FACULTY OF COMPUTER SCIENCE &

INFORMATION TECHNOLOGY

MULTIMEDIA KURONTHOGAI

by

MOHAN SELLAPPAN

WXES3182 LATIHAN ILMIAH 2

SUPERVISOR: Assoc. Prof. Dr. N. SELVANATHAN

MODERATOR: CIK HANNYZZURA AFFAL

CONTENTS

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and thanks to my supervisor Assoc. Prof. Dr. N.Selvanathan for his guidance, support and contribution to the successful completion of my project. My sincere appreciation goes to Cik Hannyzzura Affal for her constructive criticism and valuable suggestions to the further improve the project.

Thanks also go to all the lecturers who have taught me from the first year until my final year, my fellow course mates, especially Mr. Soma who has given precious guidance and encouragement throughout the project.

My appreciations and thanks also go to Mr. Sim Kian Hwa (Computer Program Technician) who provided all the necessary tools implementation of the project.

Last but not the least, this project is dedicated to my family who have given me encouragement for the successful completion of the project.

ABSTRACT

Project Multimedia *Kuronthogai* is a Final Year Project (Thesis) offered by the Faculty of Computer Science & Information Technology. This project was divided into Thesis I and Thesis II and supervised by my project supervisor Assoc. Prof. Dr. N.Selvanathan. The moderator for this project is Cik Hannyzzura Affal.

Kuronthogai is a one of the Tamil anthologies. It contains 401 poems. The theme of these poems is love and the verses are of the five lands (*tinais)* in appropriate time. Most of the Tamil literatures are in the form of a book. Only few of them are in Multimedia environment. It is important to bring the Tamil literatures in Multimedia environment. The main objective of this project is to bring the *Kuronthogai* into a Multimedia environment. This Multimedia application will be a tool that will provide almost full assistance to the Tamil literature researcher. Chapter 1 of this report covered the general view of this project. This chapter covers the objective of this project, how this project was planned, the project scheduling, expected out come and the system requirement.

Chapter 2 covers literature review . It contains a brief history of the Tamil literatures and how the poems in *Kuronthogai* were divided. The tools used for developing this application are Macromedia Director 6.0, Datagrip, Microsoft Access 7.0 and Murasu Anjal Tamil word processor. Chapter 3 covers the methodology of this application. The database design and system design were described in this chapter. Chapter 4 covers implementation, chapter 5 covers testing and chapter 6 covers evaluation.

CHAPTER 1 INTRODUCTION

Kuronthogai literally means a Tamil anthology of 401 short poems with each stanza ranging from 4 to 8 lines. This work has brought together a number of verses attributed to as many as 203 poets, of whom 15 were poetesses. The compilation of the work is attributed to a poet by name *Purikko.* The theme of this work is love and the verses are of the five lands (*tinais).* Most of the Tamil literatures in Tamil language are in form of book. Only few of them are in Multimedia environment. It is important to bring the Tamil literatures in Multimedia environment.

1.1 OBJECTIF OF THE PROJECT

The objective of this project is to bring the Tamil literatures especially *Kuronthogai* in Multimedia environment. This Multimedia application is to :

i. Bring the *Kuronthogai* in Multimedia environment

ii. Pioneer to other Tamil literatures

iii. To help understand those who can't read and write Tamil but can understand spoken Tamil

iv. To help researcher in Tamil literature

1.2 AIMS, RELEVANCE, SIGNIFICANCE OF THE PROJECT

The Multimedia *Kurunthogai* is aimed to help the Tamil researchers. It will be a tool that will provide almost full assistance to their research. This application mainly focuses into interactive Multimedia.

1.3 RESEARCH PLANS AND METHODS

To develop this multimedia application, we need to understand Tamil literature. The important part is retrieval of the key words.

1.3.1 Visiting Indian Studies Department of University of Malaya

Visiting Indian Studies Department in University of Malaya has helped to obtain information and resources about Tamil literatures especially *Kuronthogai*. Visit to Indian Studies Department includes, interview and questionnaire session with the Indian Studies Department lectures. Interview and questionnaire has enabled me to collect the related data and information.

1.3.2 Interview

Interview session will enable the Indian Studies Department lecture to provide the necessary information. They have helped us to classification of Tamil literature by means of keywords.

1.3.3 Questionnaire

Questions prepared for the questionnaire session has helped me to get more a accurate information. The questions were not technical because not all the lecture are computer literates. Questions were simple and easy to understand.

1.4 EXPECTED OUTCOME

The outcome of Multimedia *Kuronthogai* will be a computer based Multimedia application.

Two important features of the multimedia application are:

i. The application includes the lyrics and explanation in Tamil and English and the songs. The users will be able to read and listen to the songs. The users can also search the poems by searching using keywords.

ii. The application is in a multimedia environment, which contained multimedia elements like sound, text, graphics, video clips and animation.

1.5 PLANNING

Planning stage was focused on:

i. Definition of the scope of the project

ii. Project planning and scheduling

1.5.1 Scope Of The Project

Definition of scope of the project focuses on the scale of resources required. The main focus of this project is on the multimedia elements and database.

1.5.2 Project Planning And Scheduling

The development methodology for this project is System Development Life Cycle (SDLC). Each phase in SDLC is presented discretely and is never accomplished as a separate step. Several activities can occur simultaneously and activities may be repeated. Because of the cascade from one phase to another, this model is known as the 'waterfall model'. Activities were planned and divided according to planning, requirement & analysis, design, implementation, testing and maintenance.

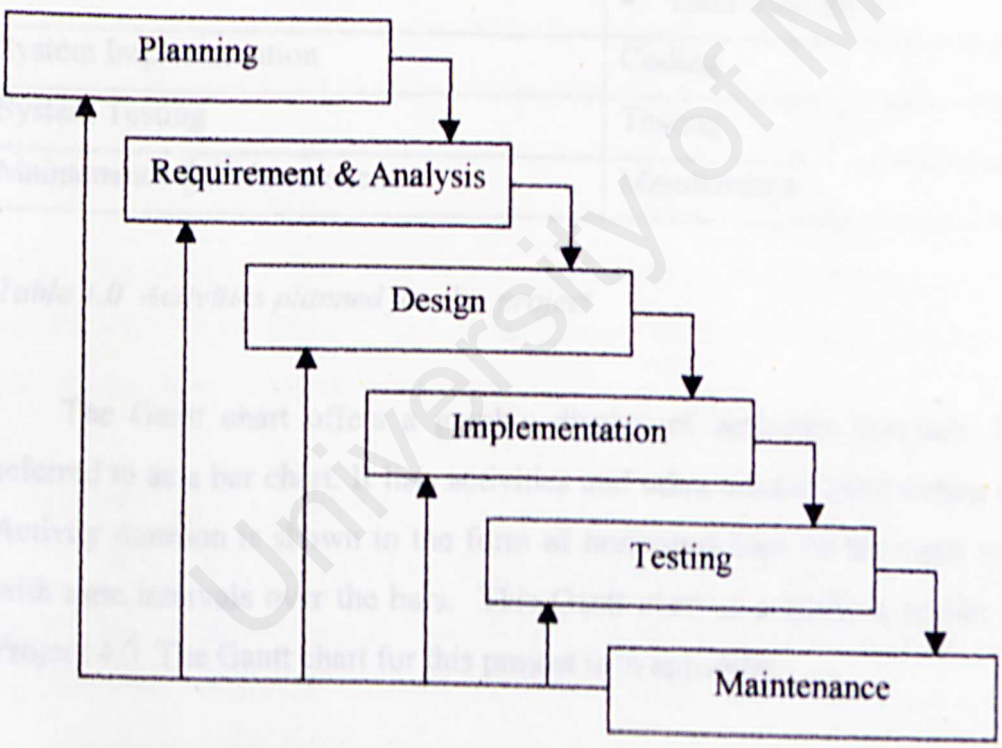


Figure 1.0 System Development Life Cycle (SDLC)

Phases	**Activities**
Planning	Preparing project proposal
Requirement & Analysis	Data collection • Interview • Questionnaire Data analysis • Data flow diagram
Design	Logical design • Design process • Design input/output • Design interface Physical design • Design database • Data dictionary
System Implementation	Coding
System Testing	Testing
Maintenance & Enhancement	Maintenance

Table 1.0 Activities planned for this project

The Gantt chart offers a graphic display of activities duration. It is sometimes referred to as a bar chart. It lists activities and other tabular information on the left side. Activity duration is shown in the form of horizontal bars on the right side of the chart, with time intervals over the bars. This Gantt chart is actually a screen from Microsoft Project 4.0. The Gantt chart for this project is in appendix.

1.6 SYSTEM REQUIREMENT

1.6.1 Software Requirement

The platform for Multimedia *Kuronthogai* is Windows 95. The tools were selected to develop Multimedia *Kuronthogai* are :

i. Macromedia Director 6.0

ii. DataGrip

iii. Microsoft Access 97

1.6.1.1 Macromedia Director 6.0

Director is an industry-standard authoring tool for multimedia production. Director not only combines multimedia elements into a portable movie, but also backs them up with Lingo, Director's own interactive scripting language. Lingo enables a Director developer and the movie's audience to control any situation in the production. Beyond Lingo and Director, if we would like to add feature to Director that Lingo doesn't provide, we can obtain or create C modules, called Xtras, which communicate with Director.

1.6.1.2 Datagrip

Datagrip is an extension Xtra for Macromedia Director and Authorware that provides a powerful and simple interface to Microsoft Access-format databases. We can use Datagrip to quickly store, retrieve, and sort all kinds of information for electronic catalogs, computer-based training, and template-based presentations. We can read and write data to Access databases, return sets of records based on an SQL query, search for records, use stored queries, and perform most of the common database operations that you can do through Microsoft Access.

1.6.1.3 Microsoft Access 7.0

Microsofrt Access 7.0 for Windows operating system provides relational database power to give information to make better decisions. It enables storage and retrieve data by Director. Microsoft Access 7.0 offers improved 32-bit performance. Efficient compilation and better data manipulation technology result in quicker response and faster data operations.

1.6.2 Hardware Requirement

The hardware and system requirements suggested for Multimedia *Kuronthogai* are:

- Windows 95 or later
- Pentium or faster processor
- hard disk with 1.2 GB space
- 32 MB of Ram
- CD ROM drive
- Mouse
- VGA or higher resolution video adopter
- Scanner
- Multimedia speaker
- Microphone

LITERATURE REVIEW
CHAPTER 2

2.1 HISTORY OF TAMIL LITERATURES

Tamil is one of the oldest living languages in the world. Other old ancient languages were Hebrew, Latin, Germanium, French, Greek etc. Almost all these classical and old languages are not spoken by the respective people now. Except for the ancient literature all of them except French perhaps are not spoken now. Somehow Tamil has survived the passage of time. It is still spoken by 80 million Tamils the world over. Having been assimilated the changes the language has grown with the ages. This adaptability of Tamil is unique in character. Tamil literature spans a period of not less than 3500 years. Future researches may even push these by a few thousand years more.

As in the case of every language the socio - political climate of the period has been affecting the growth and well being of Tamil. Tamil has been successfully emerging victorious by embracing and enabling each successive generation because of its structural perfection as well as adaptability. The literature of each period reflects the agonies and achievements, the insatiable quest for advancement in thing ephemeral and spiritual etc. of the people. Foreign invasions and overlordships have shifted the steady growth some times, but poets have arisen out of the ashes to sing the praise of the indomitable 'Mother Tamil' which continues to exude her illustrious charm till date.

The period of the three academies of Tamil covering a period of roughly 1700 years up to 250 A.D. This is considered to be the Golden Age of Tamil Literature. The literature, predominantly poetry is fresh, born out of fertile imagination takes the briefest of forms to present a picturesque panorama of the lives and times. War and love happen to be the subject matter. The serenity and chastity in love amply match nobility and greatness in war. What begins as a supreme effort of the most civilized people proceeds through steady progress to be mercilessly ended by the heartless sides of the rising

oceans, which are reported to have swallowed thousands of miles of the ebode of this great people. However zealous kings and patriotic poets have preserved whatever was possible to the benefit of the succeeding generations.

2.2 THE DIVISIONS

Tamil literature falls into three great divisions:

i. Poetry *(Iyal)*
ii. Music *(Isai)*
iii. Drama *(Natakam)*

Music and Drama have no representative works left after the great deluges. Poetry went under two distinct heads *Ilakkanam* and *Ilakkiyam*. *Ilakkanam*, which was Grammar, dealt with the art of correct and elegant writing. *Ilakkiyam* was the general term for all written works, which included all, approved poetical compositions written on the rules laid down. *Ilakkanam* was treated under five sections:

i. Letter *(Eluttu)* which constituted a part of Grammar which deals with the number, name, order, origin, form, quantity and combination of letters.
ii. Word *(Sol)* which treated of the four parts of speech, noun, verb participles and adjectives also including etymology and syntax.
iii. Matter *(Porul)* or the subject which a discourse of any kind is formed. This section treated of the passions and affections of the mind, which act internally on man and of metters belonging to the external world.
iv. Prosody *(Yaappu)*
v. Rhetoric *(Ani)*

The word *Akam* treats the 'affairs of the heart' or love between the sexes and their attitude and behavior towards each other, from the dawn of love between them, till they settle down to married life and the finals thereof.

Puram deals with all other matters of the external world, chiefly objective in nature as distinct from the subjective aspects dealt with *Akam*.

The word *Tinai* or lends itself to different interpretations such as land, class and conduct. *Akattinai* and *Purattinai* denote the behavior pattern or course of conduct of humanity.

Akattinai is treated under seven heads:

i. Unilateral Love *(Kaikkilai)*
ii. Proper Love (Union, Separation, Patience in separation, Bewailing and Sulking) *(Aintinai)*
iii. Improper Love *(Peruntinai)*

2.3 KURONTHOGAI

In all this love poetry, there are three divisions of:

i. *Mutalporul*
ii. *Karupporul*
iii. *Uripporul*

Mutal, Karu and *Uri* of each region are enunciated by *Tolkappiyar, Mutalporul* consists of land (space) which is divided into five tracts and time which is split up into *Perumpoluthu* the seasons of the year and *Sirupoluthu* or day watch and night watch.

The five divisions of land are:

i. Postoral regions *(Mullai)*
ii. Hills *(Kurinci)*
iii. Arable lands *(Marutham)*
iv. Coastal belts *(Neital)*
v. Dreary waste *(Palai)*

The six divisions of the year (August – July) are:

i. Cloudy *(Kaar)*
ii. Cold *(Koothir)*
iii. Early dew *(Munpani)*
iv. Later dew *(Pinpani)*
v. Spring *(Elaveni)*
vi. Summer *(Muthuveni)*

The six divisions of the day are:

i. Morning *(Kaalai)* 6.00 a.m. – 10.00 a.m.
ii. Noon *(Nanpagal)* 10.00 a.m. – 2.00 p.m.
iii. First hour of night *(Maalai)* 2.00 p.m. – 6.00 p.m.

iv. Evening *(Yedpadu)* 6.00 p.m. – 10.00 p.m.

v. Midnight *(Yamam)* 10.00 p.m. – 2.00 a.m.

vi. The small hours of night *(Vaikarai)* 2.00 a.m. – 6.00 a.m.

Karupporul compries the:

i. Deities *(Deivam)*

ii. Food *(Unavu)*

iii. Fauna *(Vilanggu)*

iv. Flora *(Maram)*

v. Birds *(Pul)*

vi. Drum *(Parai)*

vii. Occupation *(Seitolil)*

viii. Lyre *(Yaal)*

ix. Tune *(Pan)*

Uripporul or proper love activity is the particular episode in a man's love live, psychologically and idealistically ascribed to the particular region. The five phases of requited love and its concomitants are linked to the five fold divisions of land.

The five divisions of love are:

i. Union *(Punarthal)*

ii. Patience in separation *(Iruttal)*

iii. Sulking *(Oodal)*

iv. Bewailing *(Iranggal)*

v. Separation *(Pirital)*

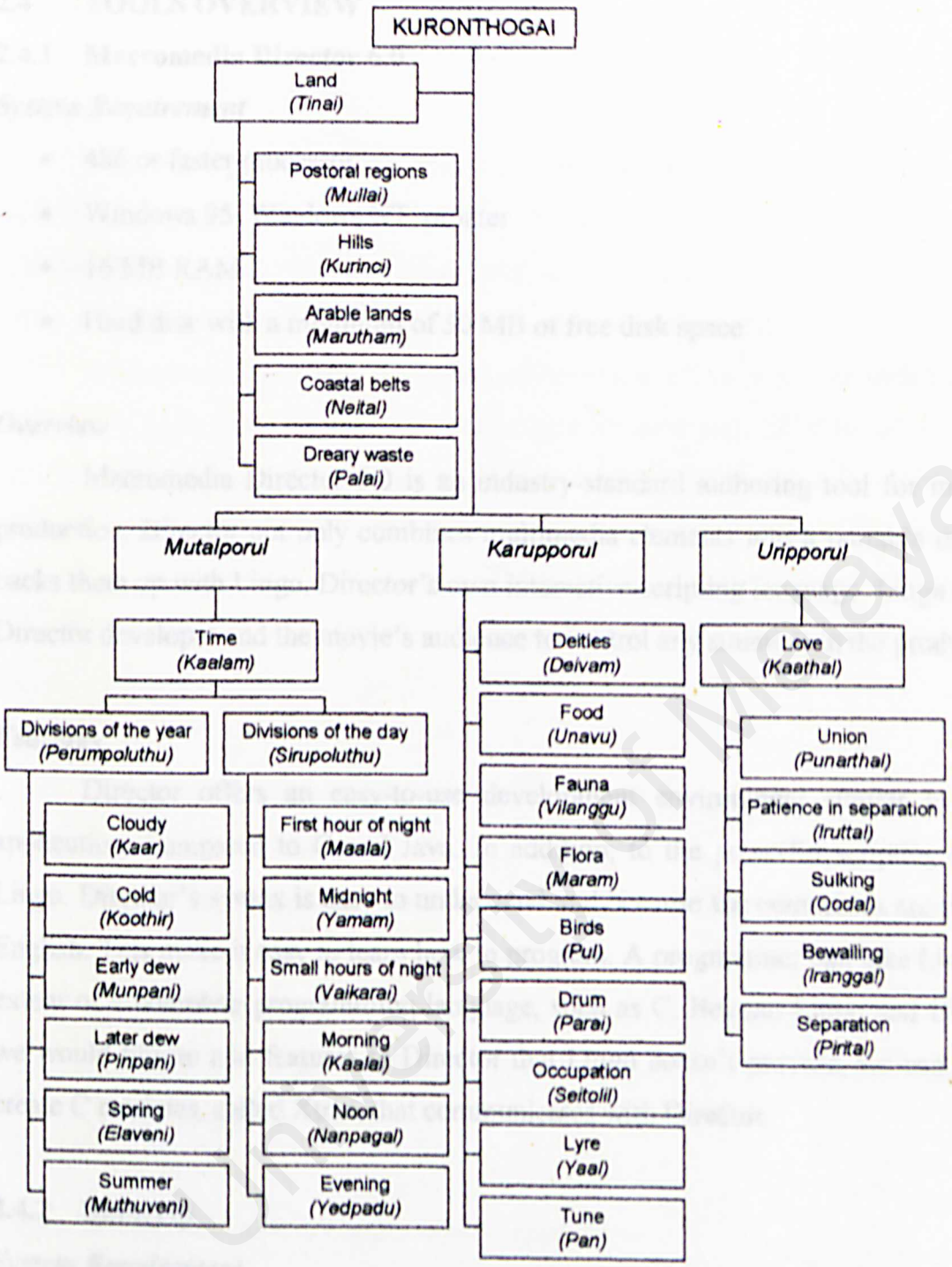


Figure 2.0 a hirercial chart of Kuronthogai

2.4 TOOLS OVERVIEW

2.4.1 Macromedia Director 6.0

System Requirement

- 486 or faster processor
- Windows 95, Windows NT, or later
- 16 MB RAM
- Hard disk with a minimum of 50 MB of free disk space

Overview

Macromedia Director 6.0 is an industry-standard authoring tool for multimedia production. Director not only combines multimedia elements into a portable movie, but backs them up with Lingo, Director's own interactive scripting language. Lingo enables a Director developer and the movie's audience to control any situation in the production.

Features

Director offers an easy-to-use development environment similar to standard applications compared to C and Java. In addition, to the powerful scripting language, Lingo. Director's syntax is easy to understand and because the commands are like actual English. This make it easy to learn how to program. A programmer can take Lingo to the extent of a complete programming language, such as C. Beyond Lingo and Director, if we would like to add features to Director that Lingo doesn't provide, we can obtain or create C modules, called Xtras, that communicates with Director.

2.4.2 Datagrip

System Requirement

- Windows '95, Windows '98 or NT 4.0 or later
- Macromedia Director version 5.0 (or later) or Macromedia Authorware 4.0 (or later)
- A 3.5" 1.44 MB floppy disk drive (or Internet connection for downloading)
- At least 5 MB of free disk space

- Microsoft Access

Overview

Datagrip is an extension Xtra for Macromedia Director and Authorware that provides a powerful and simple interface to Microsoft Access-format databases. We can use Datagrip to quickly store, retrieve, and sort all kinds of information for electronic catalogs, computer-based training, and template-based presentations.

We can read and write data to Access databases, return sets of records based on an SQL query, search for records, use stored queries, and perform most of the common database operations that you can do through Microsoft Access. In addition, since Datagrip is fully compatible with Microsoft Access version 1.0 through Access '97, we can use Access to create, edit, and browse the databases we use with Datagrip. We can even share a single database between multiple Director and Authorware programs.

Features

- Full access to Microsoft Access databases from the scripting language
- Compatible with Microsoft Access databases from version 1.0 through Access 97
- Support for stored queries allows you to store SQL queries in the database
- Support for parameterized stored queries
- Unlimited number of records and record sets (limited only by memory)
- Read and write functions for the Windows registry.
- Simple to use

Using Datagrip to create a cross-platform application

Getagrip is a cross-platform application: it runs on Mac's and PC's. Since Datagrip's run-time engine will only work on 32-bit Windows machines, it possible to create such a cross-platform application because Getagrip was developed on a PC and Datagrip was used as a build tool.

A build tool is a piece of software that runs only on the developer's system. When it runs, it creates data in another format that is used in the distribution of the application. A good example would be the build tool that is built into Director. The build tool takes

Director source as input and produces projectors as the output. In Getagrip's case, the build tool reads information from a database parses the data, then uses the new data to create scripts and text cast members.

The first and most important piece is the database. In the Getagrip application, there are six major categories: Authorware Xtras, Director Xtras, listserv information, user group information, web resources, and books. Each of these categories has a table in the database where information is kept. Let's look into the Authorware Xtra section. The table for the Authorware list has four fields: the Xtra name, a description, the company that produced the Xtra, and finally an URL to the web site to download the Xtra. The build tool looks into this table and, for each record, retrieves the Xtra's information. This information is then used to create an entry in a script. This new script is actually very interesting - a script is writing another script. To do that, a string is created with the function name, and then the data pulled from the database is used as the arguments. The new string is then assigned as a script of a cast member. Now you have a function that gets called with new data straight from the database.

In our build tool, we first create an object that has a member for each of the fields in the database. When the build tool is called, it writes a script that calls the add function of our object with the data that was written in our new script. Using this method, the entire contents of the database are stored in memory in an object-oriented fashion, making it unnecessary to use Datagrip as a run-time engine. This means the application can be shipped as is; no plugins are required. Our build tool next creates cast members that hold a text version of the record. This text version is what's visible when scrolling through the fields in the Getagrip application. Now that there is a version of the data in text fields and a script that will load it into memory at any time, a user interface can be designed as if Datagrip was actually running in the background. The following is a visual representation of the process.

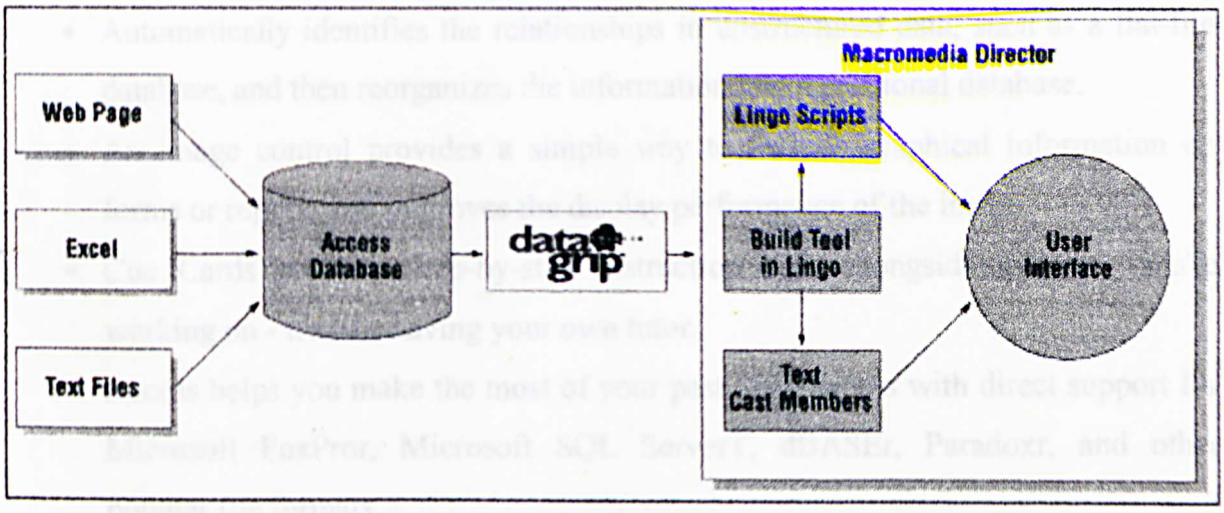


Figure 3.0 visual representation of the datagrip

2.4.3 Microsoft Access 7.0

System Requirement

- 486 or faster processor
- Windows 95, Windows NT, or later
- 16 MB RAM
- Hard disk with a minimum of 40 MB of free disk space

Overview

Microsoft Access 7.0 for the Windows95 and Windows NT operating systems provides relational database power to give you the information you need to make better decisions. It integrates data from spreadsheets and other databases, and is the easy way to find answers, share information over Intranets and the Internet, and build faster business solutions. Access 7.0 allows you to generate, analyze and create reports without hours of work. It integrates ease of use from the data entry point to printing in HTML.

Feature

- Automatically builds tables, queries, forms, and reports from more than 20 types of full-featured templates, giving you the option to further customize them to suit your needs.

- Automatically identifies the relationships in unstructured data, such as a flat-file database, and then reorganizes the information into a relational database.
- An image control provides a simple way to include graphical information on forms or reports and improves the display performance of the image.
- Cue Cards provides step-by-step instruction right alongside the task you're working on - it's like having your own tutor.
- Access helps you make the most of your past investments with direct support for Microsoft FoxPror, Microsoft SQL ServerT, dBASEr, Paradoxr, and other popular file formats.

2.4.4 True Type Font

True Type Font is a popular form of scalable outline font that will work with most printers and is also able to render appropriately on screen. *Murasu Anjal* Tamil True Type Font will use to develop this application.

METODOLOGY
CHAPTER 3

3.1 DATABASE DESIGN

Database is a collection of information stored in an organized form in a computer. The design of the database model will support me to do this application. The major aims of database design are:

- To represent the data and the relationship
- To provide a data mode
- To specify a design that will achieve the stated performance requirements for the application such as response time

Database design is divided into two phases:

i. Logical Database Design

ii. Physical Database Design

3.1.1 Logical Database Design

Logical Database Design is a process of constructing a model of the information use in an enterprise based on one model of data. In the external schema the user will perceive the data and the conceptual schema will describe:

- *what* data is stored in the database
- entities, their attributes and their relationships among the data.
- the constrains on the data
- semantic information about the data
- security and integrity information

The tables below shows the design view of the tables in *Kuronthogai* database.

1) MAIN

Field Name	Data Type	Field Size	Description
PoemNo	Number	Long Integer	Poem number
Title	Text	50	Title of poem
TitleSortIndex	Number	Long Integer	Title sorting index
FirstLine	Text	50	First line of the poem
FirstLineSortIndex	Number	Long Integer	First line sorting index
Poem	Memo	400	The lyric of the poem
Explanation	Memo	200	The explanation of the poem
Land	Text	15	Land type
Poet	Text	50	The poet of the poem
Sender	Text	15	The message from
Target	Text	15	The message to

2) YEAR

Field Name	Data Type	Field Size	Description
YearDivision	Text	15	The division of the year
PoemNo	Number	Long Integer	Poem number

3) DAY

Field Name	Data Type	Field Size	Description
DayDivision	Text	15	The division of the day
PoemNo	Number	Long Integer	Poem number

4) DEITIES

Field Name	Data Type	Field Size	Description
Deities	Text	15	The deities
PoemNo	Number	Long Integer	Poem number

5) FOOD

Field Name	Data Type	Field Size	Description
Food	Text	15	The foods
PoemNo	Number	Long Integer	Poem number

6) FAUNA

Field Name	Data Type	Field Size	Description
Fauna	Text	15	The fauna
PoemNo	Number	Long Integer	Poem number

7) FLORA

Field Name	Data Type	Field Size	Description
Flora	Text	15	The flora
PoemNo	Number	Long Integer	Poem number

8) BIRD

Field Name	Data Type	Field Size	Description
Bird	Text	15	The birds
PoemNo	Number	Long Integer	Poem number

9) DRUM

Field Name	Data Type	Field Size	Description
Drum	Text	15	The drums
PoemNo	Number	Long Integer	Poem number

10) OCCUPATION

Field Name	Data Type	Field Size	Description
Occupation	Text	15	Their occupations
PoemNo	Number	Long Integer	Poem number

11) LYRE

Field Name	Data Type	Field Size	Description
Lyre	Text	15	The lyre
PoemNo	Number	Long Integer	Poem number

12) TUNE

Field Name	Data Type	Field Size	Description
Tune	Text	15	The tune
PoemNo	Number	Long Integer	Poem number

Entity relationship model, views entity types, relationship type and attributes. The entity relationship model for this database application is in the appendix.

3.1.2 Physical Database Design

Physical Database Design is a process of producing a description of the implementation of the database on secondary storage; it describes the storage structures and access methods used to effectively access data. Physical database design was stared the completion of the logical database design.

3.2 SYSTEM DESIGN

3.2.1 Multimedia

Multimedia is a combination of hardware and software that can produce output that combines text, graphics, sound, video clips and animation along with controls that allow users to navigate through the finished multimedia application. The success of a Multimedia Application depends on the skillful use of multimedia elements.

Text

Text material is the fundamental form of communication. Viewers will read vast quantities of textual material when reading a book or newspaper, yet will not read lots of text on a screen. Murasu Tamil True Type Font will be used to view the textual material.

Graphics

Graphics are the strongest component of Multimedia . It is hard to envision a Multimedia Application without a liberal use of icons, photographs, or artwork.

Graphics are used to:

- add color and images to the page background
- enhance spatial design to the page layout
- communicate personal style
- display graphic artwork and photographs
- display spreadsheets and graphs
- create text graphics for layout effects.

All the graphics and images that is used will use in this application especially the background are related to the poem so as the enable the user to see the right context. Indian classical instruments will be represented bye suitable buttons and clickable images.

Sound

Sound is a potent communication medium. We can use sound effects, music and narration to enhance the multimedia application. Sound files such as *.au, *.wav *.aif formats tend to be large depending on the quality of the sound desired. The Real Audio format *.ra compresses the file to one-hundredth the original size with reasonable quality and supports streaming audio .

Sounds are used to:

- support visuals
- attract attention and interest

- improve perceptions of quality
- represent things not shown
- provide feedback

Recording session will be held with singers. The sound files will be added to this application and the users are able to listen to the song when they read the lyric of the poem. Indian classical music will be added to the application as a background sound.

Video

Video is the most powerful and most expensive medium for online documentation. It is ideal for showing images of real moving objects. Producing video requires specialized equipment, an extensive team of expert professionals and an ample budget. For interactive video, design costs and times are about seven to ten times those of simple video production and require over 300 hours of authoring, for each hour of interactive video.

Video is used to:

- show how things move
- show what users cannot see directly
- motivate users
- show human behavior and emotions

Video clips will not be added in this application.

Animation

Animation is the illusion of motion. Animation tends to amplify whatever else is occurring in terms of visual impact. Full motion and sound requires the use of such large files as *.fli *.flc or *.avi. There are 2D and 3D programs such as 3DStudio, Truespace, Alias that

create full motion using computer generated images. However in Macromedia Director 6.0 it self can create animations. Animated GIF Images are created by sticking together many *.gif images that appear in rapid succession, in the same location giving the sense of illusion. Animations will be added to this application to make this application more attractive.

3.2.2 User Interface Design

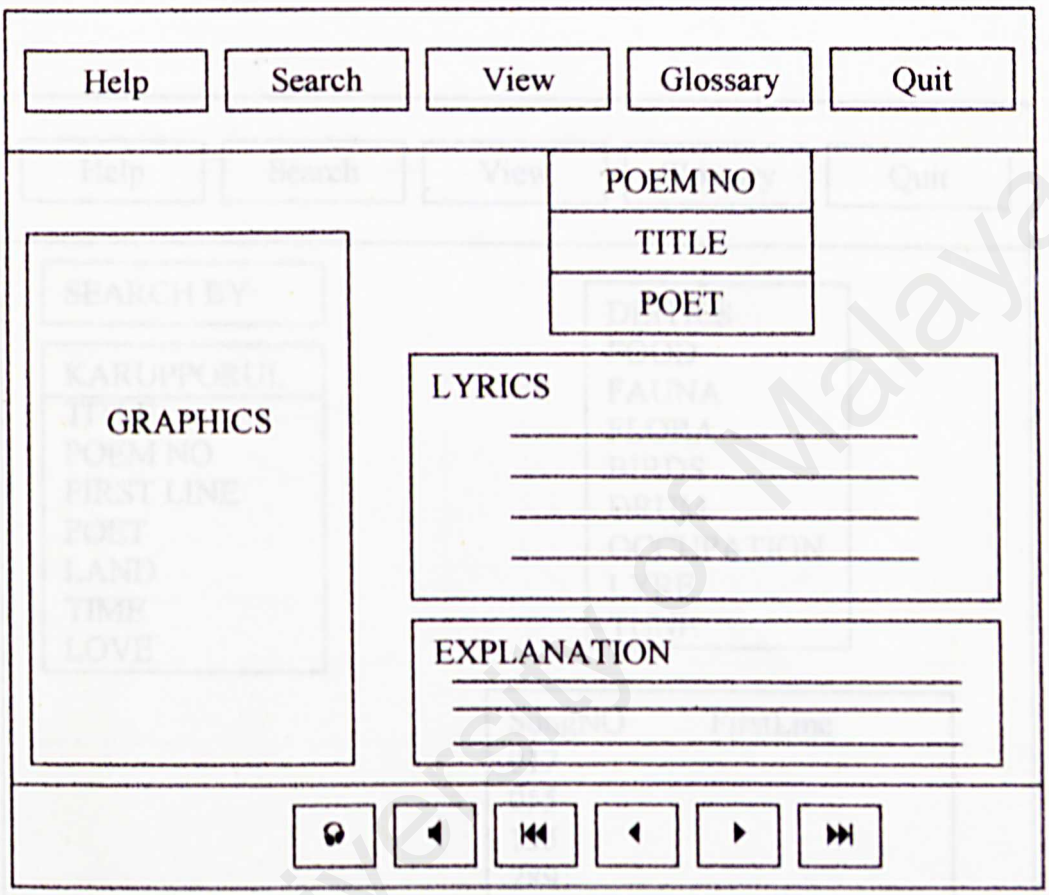
Before implementing a form, it is essential that we first design the layout. The list of guidelines for the form designing are (Shneiderman, 1992):

- Meaningful title
- Comprehensible instructions
- Logical grouping and sequencing of fields
- Visually appealing layout of the form
- Familiar field labels
- Consistent terminology and abbreviations
- Consistent use of colors
- Visible space and boundaries for data-entry fields
- Convenient cursor movement
- Error correction for individual characters and entire fields
- Error message for unacceptable values
- Optional fields marked clearly
- Explanatory messages for fields
- Completion signal

The forms below are user interface design for this application.

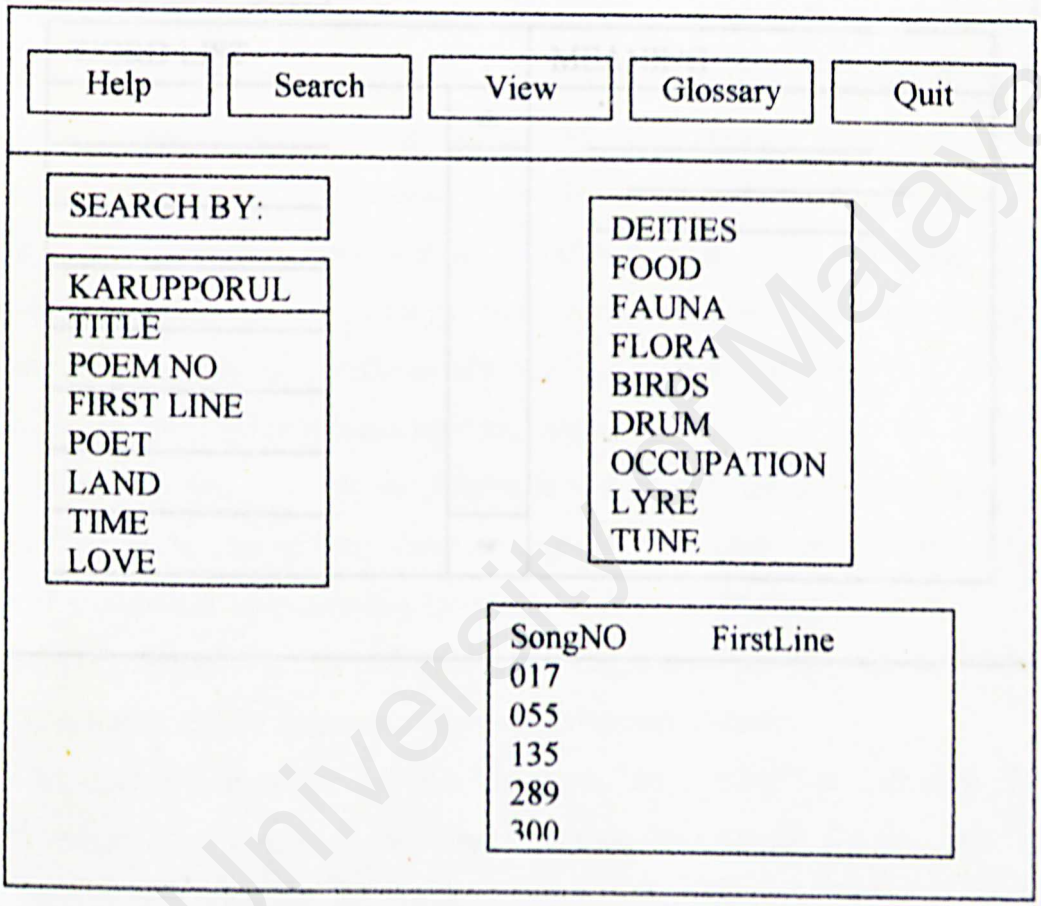
Form 1

This form shows the lyrics of the poem, the poem's title, poem number, explanation, poet and related animation or graphics. This form also has buttons like ear phone, sound on off, go to first poem, go to previous poem, go to next poem and go to last poem.



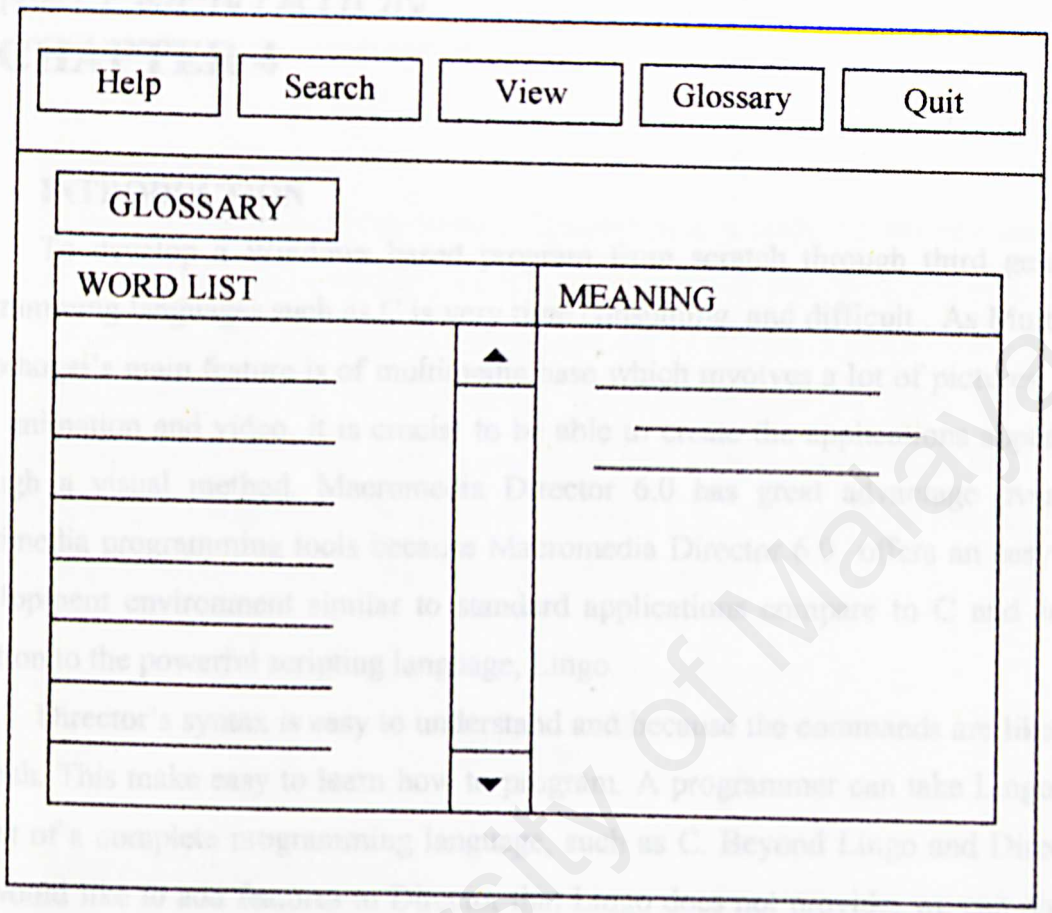
Form 2

This form shows the search how a poem was searched according to *Karupporul*, title, poem number, first line of the poem, poet, land, time and love. This form shows an example how was the poem number 289 selected from the *Karupporul* birds.



Form 3

This form shows the glossary and the meaning.



IMPLEMENTATION
CHAPTER 4

4.1 INTRODUCTION

To develop a Windows based program from scratch through third generation programming languages such as C is very time consuming, and difficult . As Multimedia Kurothogai's main feature is of multimedia base which involves a lot of pictures, sound, text, animation and video, it is crucial to be able to create the applications appearances through a visual method. Macromedia Director 6.0 has great advantage over other multimedia programming tools because Macromedia Director 6.0 offers an easy-to-use development environment similar to standard applications compare to C and Java, in addition to the powerful scripting language, Lingo.

Director's syntax is easy to understand and because the commands are like actual English. This make easy to learn how to program. A programmer can take Lingo to the extent of a complete programming language, such as C. Beyond Lingo and Director, if we would like to add features to Director that Lingo does not provide, we can obtain or create C modules, called Xtras, that communicates with Director.

Macromedia Director 6.0 has extensive error handling facilities. There are literally hundreds of possible errors that Macromedia Director 6.0 can catch and deal with. Macromedia Director 6.0 handles design time errors by insisting that they are corrected before running the program. When logical errors crop up, it's debugger enables the programmer to find out what went wrong by working through the program, statement by statement if a need arises. The debugging tools include the Debug window, breakpoints, single stepping and watch.

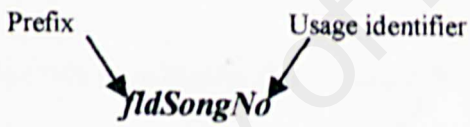
4.2 CODE DOCUMENTATION

Code documentation is a set of written descriptions that explains to a reader what the programs do and how they do it. Internal documentation is descriptive material

written directly within the code. All other documentation is external documentation. Code documentation begins with the selection of identifier (variables and labels) names, continues with commenting and ends with the organization of the program. Typically, keeping track of and maintaining an existing code base consumes the most time and effort in most of the programming projects.

4.2.1 Naming Conventions

In the development of Multimedia Kuronthogai, a standard naming convention was used to increase the readability of the code. Naming standards can help prevent costly and embarrassing mistakes. By adopting a set of standards and sticking to it, any future enhancements made to Multimedia Kuronthogai can be done without any misunderstanding and confusion. Each control in Multimedia Kuronthogai has a unique prefix that tells the programmer what kind of control it is. This prefix is followed by the usage identifier which describes the function of the controls



4.2.2 Internal Documentation

Internal documentation contains information directed at the person who will be reading the source code of the program and might possibly enhance the application. Thus in Multimedia Kuronthogai, summary code is provided to identify the program and a describe its data structures, algorithms and control flow. A statement of purpose dictating the function of the module and descriptive comments are embedded within the body of the source code to describe processing functions. In Macromedia Director 6.0, the readability of the comments is enhanced as the color of the comments differs from the color of the program codes. The symbol for the comments in Macromedia Director 6.0 is double hyphen (--)

An example of using comments in the coding of Macromedia Director 6.0:

This function set the appropriate length so that it will display correctly in a 2 column list box:

```
on PadText str
        --Set the num of chars before the next column
        set padLength = 20
        set count = length(str)
        if count <= (padLength - 1) then
                repeat with i = 1 to (padLength - count)
                        set str = str & " "
                end repeat
        else
                set str = chars(str, 1, padLength - 1) & " "
        end if
        return str
end
```

4.3 WORKING WITH MACROMEDIA DIRECTOR 6.0

4.3.1 Cast

A cast is a library of graphics, sounds, color palettes, behaviors and Lingo scripts, buttons, transitions, digital video movies, and text used in a Director movie. Each movie's cast can contain up to 32,000 cast members.

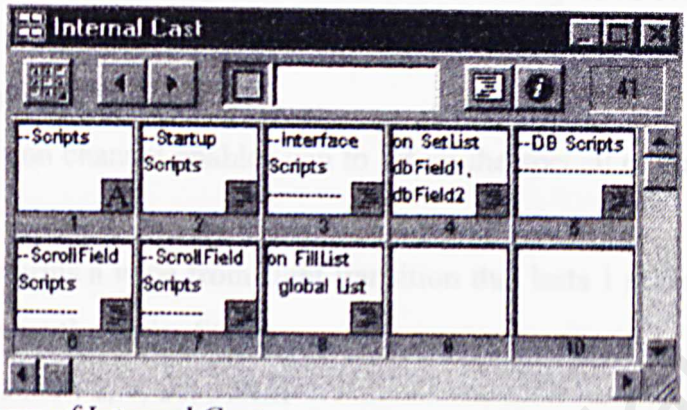


Figure 4.1 a diagram of Internal Cast

4.3.2 Score

The most important Window in Macromedia Director 6.0 is the score. It's similar to a spread sheet – the columns are called frames, and they correspond to relieve instances of time; the rows are called channels, and they correspond to types of content.

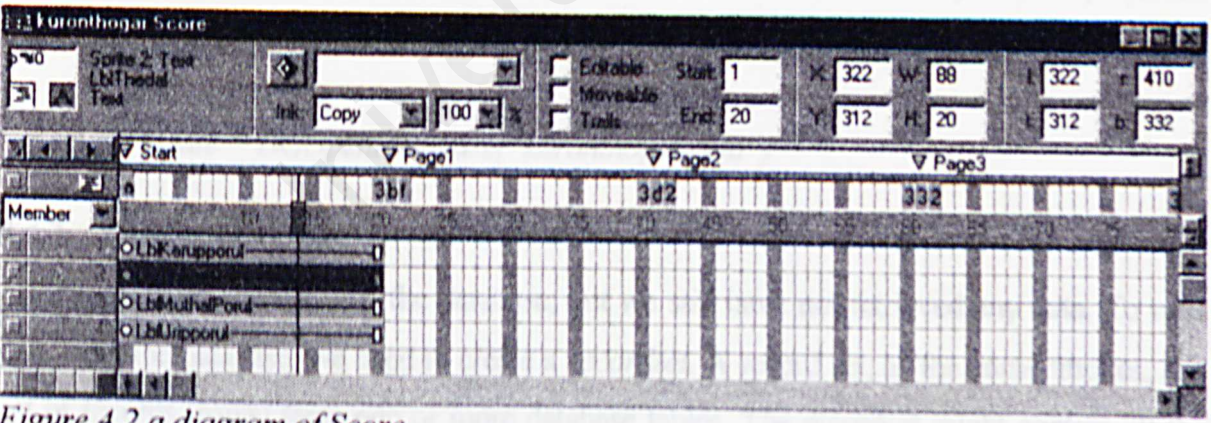


Figure 4.2 a diagram of Score

4.3.3 Ink Effect

Ink Effect applied to the sprites to change the way they appear on the Stage. The Ink pop up also indicates the current ink applied to selected sprites in the Score. The syntax for setting the ink of a sprite with Lingo is:

set the ink of sprite 3 to 36

This example sets the ink of the sprite in channel 3 to Background Transparent.

4.3.4 The Transition Channel

The Transition channel enables you to set up the special effects among the scenes of your movie.

This statement performs a wipe from right transition that lasts 1 second, has a chunk size of 20, and occurs over the entire Stage:

puppetTransition 2, 4, 20, TRUE

4.4 WORKING WITH DATAGRIP FUNCTIONS

4.4.1 Open Database

The first step in just about any database application is to open the database file. You'll use the **DGOpenDatabase** function. DGOpenDatabase takes one parameter, the name of the database you want to open. If the database file exists in the same directory as your projector, you can simply use the name of the file like this:

set dbHandle = DGOpenDatabase("kuronthogai.mdb")

4.4.2 Create Recordset

Even though we opened the database, but we still have not retrieved any data. To read data from the database, we need to create a *recordset.* A record set, as the name implies, is a set of records (rows) from one or more database tables. The recordset might contain all the records in the table, or it might contain only a few records, based on some criteria we specify in an SQL *query.* Here's what a typical call to **DGCreateRecordset** looks like:

*set rsHandle = DGCreateRecordset("SELECT * FROM tblMain", dbHandle)*

4.4.3 Working With QueryDefs

A QueryDef is simply a query, just like the ones we've seen up to this point, stored in the database file and given a name. The advantage of stored queries is that they are faster than query strings in your code. If you use a particular query several places in your code, it's also easier to maintain a single query in the database than defining it all the places you need it in your Lingo code.We open a QueryDef using the **DGQDOpen** function. Here's an example:

Set qdHandle = DGQDOpen("qrPoet", dbHandle)

The first parameters to **DGQDOpen** is the name of the stored QueryDef. The second parameter is the database handle returned from **DGOpenDatabase**.

4.4.4 Parameterized Queries

We'll store this query in the database as a QueryDef. However, we need to supply a value for the SongNo variable each time we use the query. This is called a "parameterized query" because we store the query. Here's a segment of code that illustrates the process:

```
set val = DGRSGetFieldValue(dbfield1, rsHandle1)

 if val <> "#ERROR#" then

  DGQDSetParameterValue("fldID",identifier, qdHandle1)

  set tmpRS = DGQDCreateRS(qdHandle1)

end if

set val = DGRSGetFieldValue("ContactID", rsHandle)

if val <> "#ERROR#" then

  DGQDSetParameterValue("pID", val, qdHandle)

  set tmpRS = DGQDCreateRS(qdHandle)

end if
```

In the first line we used a new Datagrip function to read the value of the dbfield1 field from any table using a recordset we have previously created. As long as no error occurred in reading the value, we use that value in a **DGQDSetParameterValue** function. This function takes three parameters:

fldID is the name of the parameter in the query that we want to set.

identifier is the value that we are setting for the parameter

qdHandle is a handle to a previously opened QueryDef.

4.5 WORKING WITH TAMIL EDITOR

There are a number of Tamil fonts that we can use to edit tamil text such as Murasu, Nalinam, TamilOssai and so on. Among all these, Murasu Tamil Font is popular because it is easy to use and it has different type of fonts. We have to use a dongle to view the tamil fonts. The fonts must be typed in the editor first and then cut from the editor and pasted to the field in Macromedia Director 6.0.

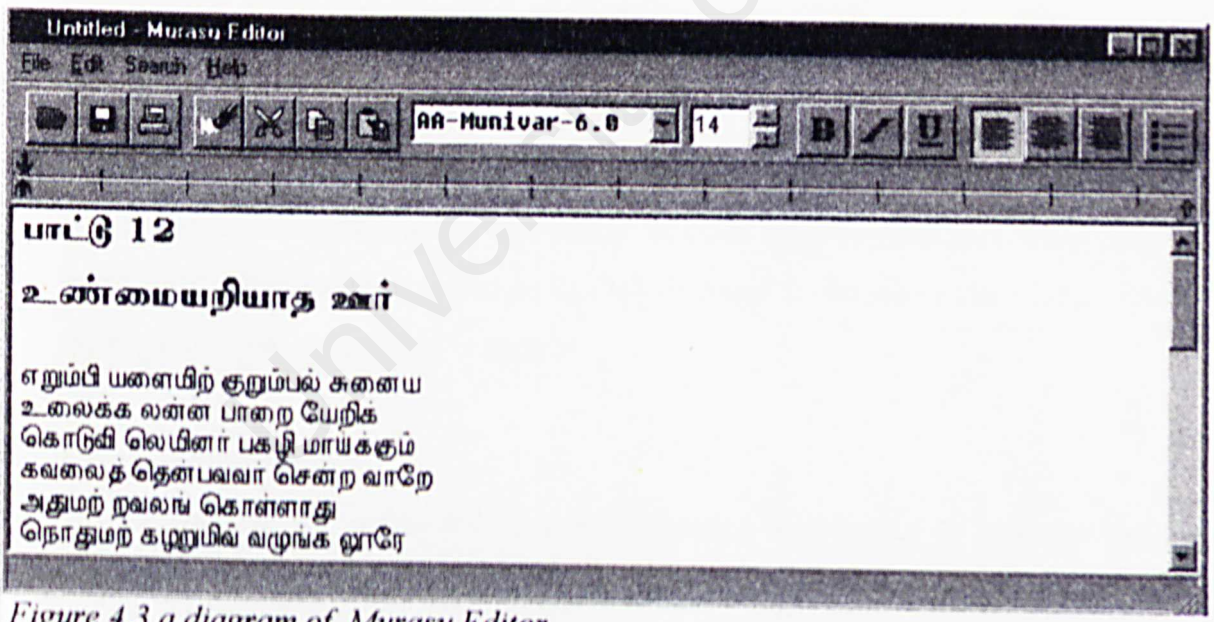


Figure 4.3 a diagram of Murasu Editor

4.6 WORKING WITH MICROSOFT ACCESS

Microsoft Access is a very user friendly tool that helped to a great extent the completion of this thesis. The data must be typed in Murasu Tamil editor before it can be stored in tables in Microsoft Access.

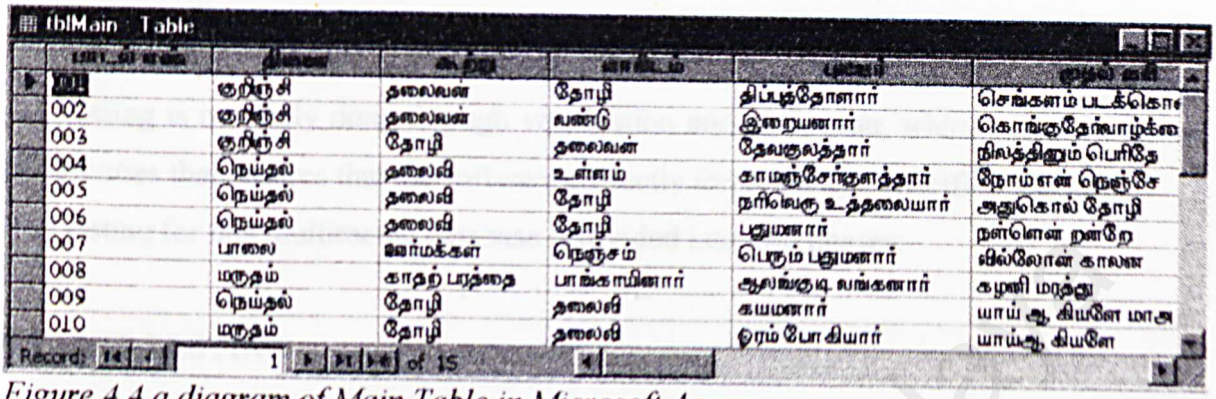


Figure 4.4 a diagram of Main Table in Microsoft Access

TESTING
CHAPTER 5

Testing is normally done through verification and validation, which refers to the set of activities that ensures that the software correctly implements a specific function. The testing for this multimedia database is divided into two phases:

5.1 UNIT TESTING

It focuses on the individual components. These individual components especially individual templates, are tested to ensure that they operate correctly. Each component or form is tested independently, without involving other system components. This enables errors to be detected in coding and logic that are contained within the function alone.

a) Interface

The module interface in this application is tested to ensure that information properly flows in and out of the program unit under testing. For example, testing will be conducted to ensure that the data that is received from another part of the program. Interface testing also tests to ensure that the data that flows out of the module into the other parts of the application is correct.

b) Local Data Structure

The unit testing conducted on this application also serves to examine the data structure of the module to ensure that the data stored in the module temporarily maintains its integrity during the execution of the module.

c) Independent path testing.

All branches of the module's coding in this application are executed at least once to see whether it works as it should.

d) Boundary conditions

It ensures that the module operates properly at the boundaries established to limit or restrict its processing.

e) Error handling

Testing of the error handling capabilities of the modules in this application. The module should be able to detect and recover from any error that occurs during its execution.

5.2 INTEGRATION TESTING

Once unit testing has been completed, all the individual units are combined into a working system. The integration is planned and coordinated so that when an error occurs, there is some idea of where the error could have occurred. Integration testing is to check for systematic approach for constructing the program structure while simultaneously conducting tests to uncover errors caused by the interfacing. The primary concern is that the system meets its functional and non-functional requirements and also to reduce time in looking for errors.

Integration testing for this application was done using the bottom-up approach. Components at the lowest level of the hierarchy is tested individually, first and then later all the individually tested components are jointly tested. This approach was done repeatedly until all the components are tested. This system allows faults to be discovered in each unit before combining them, which facilitates the tracking of faults when they occur.

5.3 USER ACCEPTANCE TESTING

The user acceptance survey was conducted to find out whether Multimedai Kuronthogai was successfully implemented and accepted by the users. The users comprised of two undergraduates from Faculty of Arts.

The students found the system to be user friendly. The application is simple and nice especially the backgrounds, easy to search according to the keywords of song

numbers, poet and land. The feedback from the evaluation was that the project covered be further enhanced by including more animations and video clips.

The following suggestions were given by the users to improve Multimedia Kuronthogai:-

1. should include more animations and video clips.
2. should include statistic column
3. should improve the beginning part

SYSTEM EVALUATION
CHAPTER 6

System evaluation, is the process of identifying system strength and limitations by measuring the system being built against expectations. Problems are encountered and most of them were resolved eventually.

6.1 SYSTEM EVALUATION

6.1.1 System Strength

Evaluation of system strength is focused on the strength of the database . This is because all the data are accessed by Macromedia Director 6.0 interface from the database in Microsoft Access. By developing the system database in this project, the system has inherited the strength of being able to update with new data and not burden the Macromedia Director 6.0 applications or coding.

The system strength depends on certain criteria such as:

- **Searching capabilities**

  Multimedia Kuronthogai provide search modules like search by Song Number, Poet, Land, First Line, Song Title, Time and Matter. The users can choose what ever search method they want. In the result of search by Song Number, Poet, Land, Time and Matter will contain Song Number and First Line which was sorted by Song Number. But in the result of search by First Line and Song Title will contain Song Number and First Line but was sorted by First Line or Song Title.

- **No keyboard skills required**

  Multimedia Kuronthogai eliminates the need for any typing or alphabetizing skills. It is just a matter of pointing the mouse at a particular search field and clicking the mouse button.

- **No spelling skills required**

  Multimedia Kuronthogai eliminates the need for any typing skills.

- **Simple and User-friendly Interface**

  The multimedia database application interface was developed in a simple and user friendly manner.

- **Provide Maintenance Facilities**

  It is easy to make corrections if any errors occur.

6.2 SYSTEM LIMITATIONS

- **Retrival of data speed**

  If Multimedia Kurunthogai is implemented on Pentium processor or lower PC model with less than 32MB Ram, the retrival of information from its database is very slow.

- **Can't run from CD**

  Multimedia Kurunthogai can't run directly from CD, it must be installed in a hard disk first.

- **Limited to the resolution**

  The resolution must be 1024x768 because Macromedia Director 6.0 only can read the resolution but can not change the resolution.

- **Must use Murasu Tamil Font**

  Murasu Tamil Fonts were used in this application. To view the Tamil Fonts the users must run the Murasu Tamil Font first, before they start this application.

6.3 PROBLEM AND SOLUTIONS

6.3.1 Unfamiliarity With Macromedia Director 6.0

Although Macromedia Director 6.0 is widely considered to be one of the easiest Multimedia programming tool, there was still difficulty faced in learning on how to use Macromedia Director 6.0. The UM library do not have any books on Macromedia Director 6.0. The are only few books available in local market and the books are expensive. None of the Macromedia Director 6.0 books provided any in depth reference on using alternative methods to connect to the database. Help was sought from friends and Macromedia website in using alternative methods to connect to the database.

6.3.2 Insufficient Time

A lot of time was spent in analyzing the data that was provide by the Indian Studies Department. Time was also spent to learn the usage of the actual coding process was stared at a later stage.

6.4 FUTURE ENHANCEMENT

The development of any system is always a dynamic process. Further improvement and new ideas will help in enhancing of the performance of the system. However, due to time constraint, these ideas could not be incorporated into this system. The future enhancement for this project is to provide a English version. The English version will help those who are not Tamil educated to read this poems.

6.5 KNOWLEDGE GAINED

Knowledge has been gained during the whole process of development and implementation of the project. The knowledge gained is as follows:

- I learnt about Macromedia Director 6.0 program design and development;
- Techniques to plan and develop a multimedia system;
- Techniques to design and maintain the database;
- Learnt about other additional tool such as Adope Image Styler and Coll Edit Pro for wave editing
- I obtained experience in scanning and voice recording, data collection, problem solving and improvement of communication skill during data collection.
- Additional understanding of Tamil literatures, especially Kuronthogai

6.6 CONCLUSION

Multimedia Kuronthogai has finally met its objective. Although this is considered as the initial goal and specification, there are still requirements needed for further improvement.

Usage of Visual Basic as a multimedia development tool is a great advantage as it uses the latest concepts.

Theory and knowledge gained throughout the course of studies were applied in the development of Multimedia Kuronthogai. Although there is still room for improvement in Multimedia Kuronthogai, it can be a guide for the students who will be under taking projects like this in future.

Finally I hope that this application which is designed to give multimedia environment about Kuronthogai will be a useful application tool for Tamil literature researcher especially to the Indian Studies Department in University Malaya.

BIBLIOGRAPHY

Internet web links:

Macromedia Director 6.0

- *http://www.macromedia.com/*

Datagrip

- *http://www.datagrip.com/*

Microsoft Access 7.0

- *http://www.microsoft.com/canada/products/prodref/3_ov.htm*
- *http://agent.microsoft.com/catalog/display.asp?site=444&subid=22&pg=1*

Kuronthogai

- *http://www.intamm.com/literature/index.htm#Introduction*

Books on:

Macromedia

- *INSIDE Macromedia Director 6 with Lingo*

  Lee Allis

Microsoft Access 7.0

- *Special Edition using Access 97*

  Roger Jennings

Kuronthogai

- *A HISTORY OF TAMIL LITERATURE with Text and Translation*

  J.M. Somasundram Pillai, B.A, B.L 1967

- *A CRITICAL STUDY OF KURONTHOGAI*

  Dr. T. Leelavathy

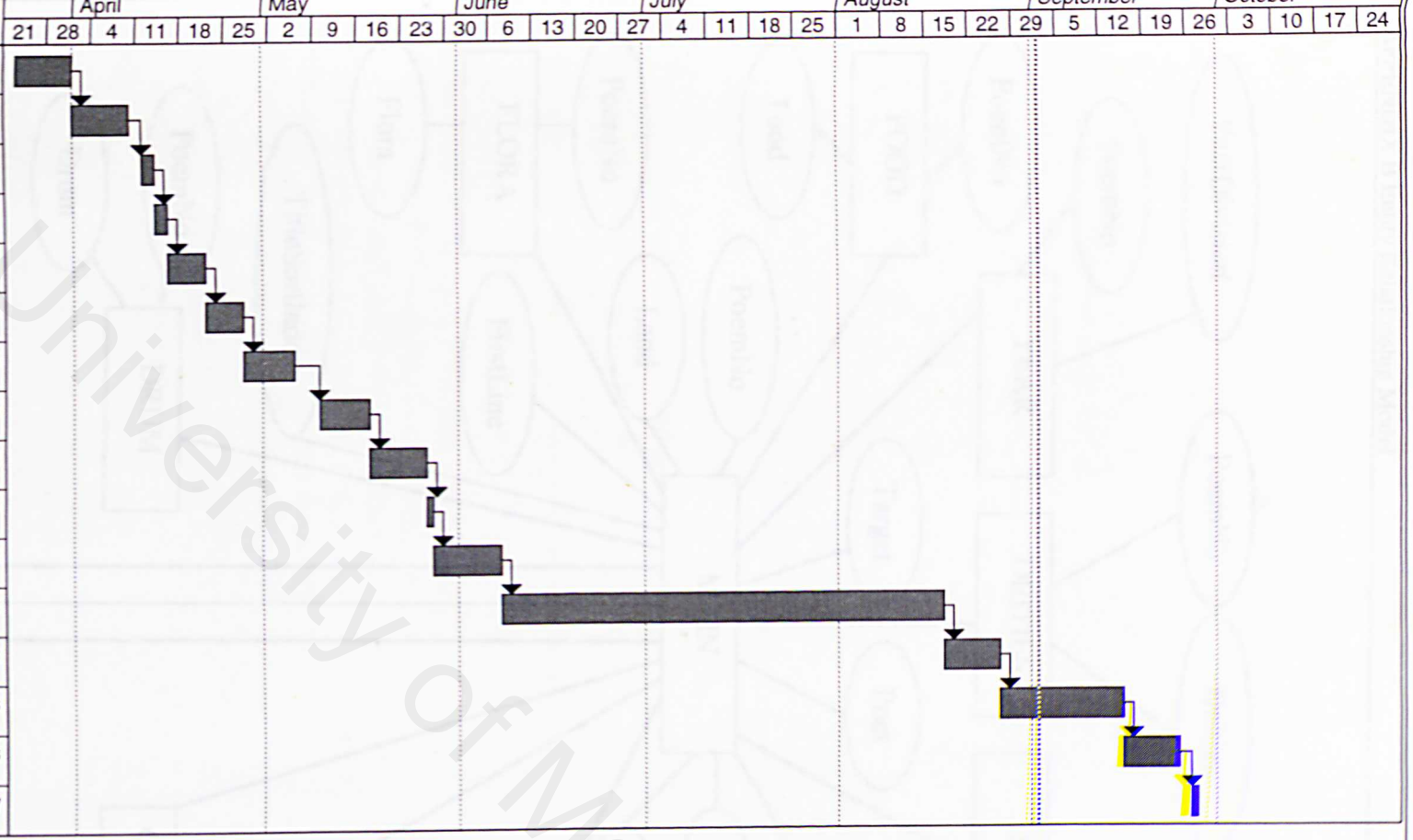
- *THOLKAPPIYA ILLAKIA KOLGAIYUM KURONTHOGAIYUM*

  Dr. K. Kalimuthu, MA, PhD.

- *KURONTHOGAI – Mulamum Uraiyum*

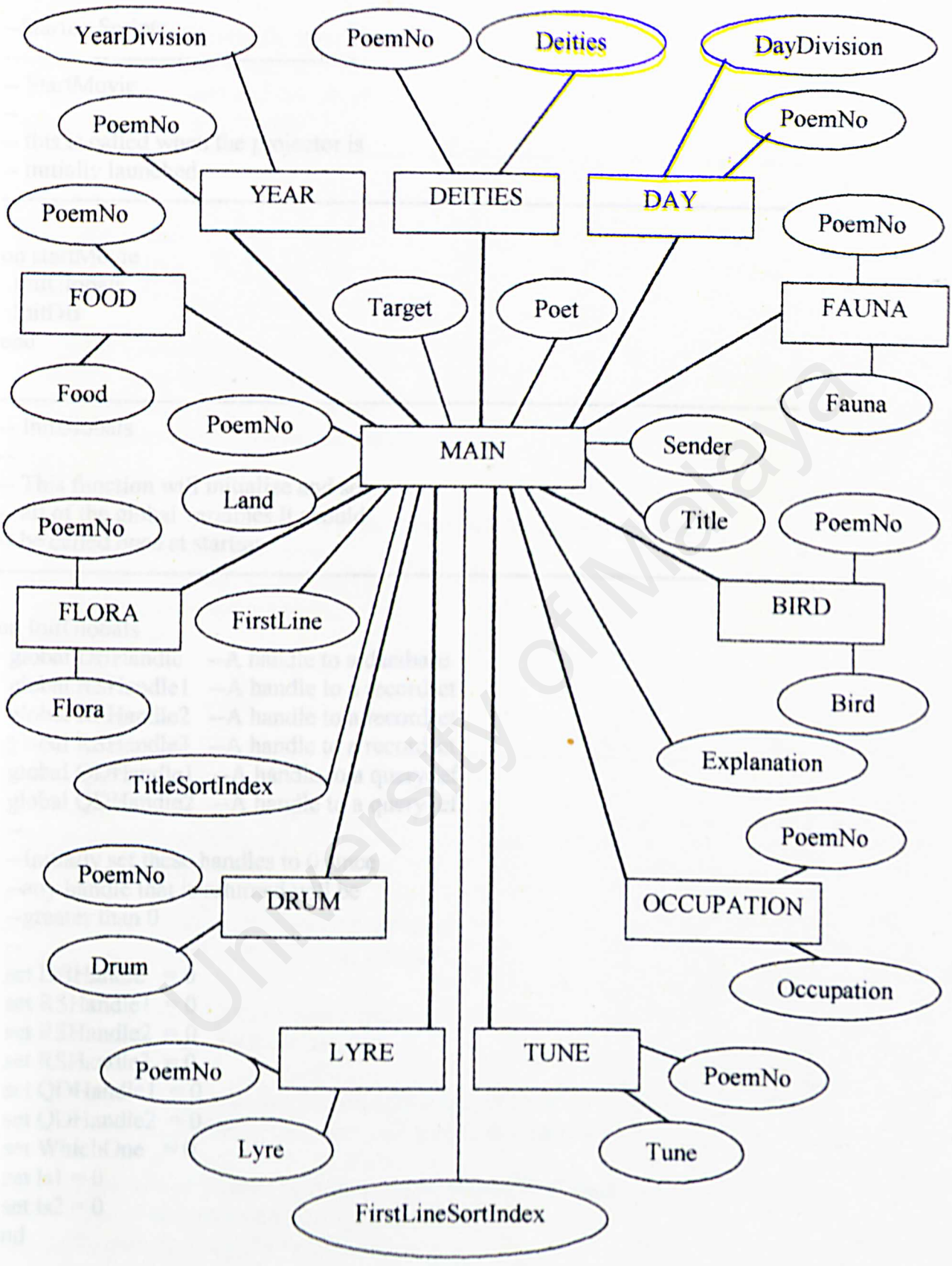
M. Shanmugam Pillai

ID	Task Name	Duration
1	Preparing project proposal	7 days
2	Data Collection	7 days
3	Interview	2 days
4	Questionnaire	2 days
5	Analysis	4 days
6	Synthesis	4 days
7	Planning	6 days
8	Documentation I	6 days
9	Logical Design	7 days
10	Viva I	1 day
11	Physical Design	7 days
12	System Implementation	51 days
13	System Testing	7 days
14	Maintenance & Enhancement	14 days
15	Documentation II	7 days
16	Viva II	1 day



Project: Project1
Date: Thu 9/2/99

Task
Split
Progress
Milestone
Summary
Rolled Up Task
Rolled Up Split
Rolled Up Milestone
Rolled Up Progress
External Tasks
Project Summary



Entity Relationship Model

CODING

```
--Startup Scripts
-----------------------------------
-- StartMovie
--
-- this is called when the projector is
-- initially launched
-------------------------------------------------------------------------------

on startMovie
  InitGlobals
  InitDB
end

-------------------------------------------------------------------------------
-- InitGlobals
--
-- This function will initialize and set
-- all of the global variables.It should
-- be called once at startup
-------------------------------------------------------------------------------

on InitGlobals
  global DBHandle    --A handle to a database
  global RSHandle1   --A handle to a recordset
  global RSHandle2   --A handle to a recordset
  global RSHandle3   --A handle to a recordset
  global QDHandle1   --A handle to a querydef
  global QDHandle2   --A handle to a querydef
  --
  --Initially set these handles to 0 since
  --any handle that is returned will be
  --greater than 0
  --
  set DBHandle  = 0
  set RSHandle1 = 0
  set RSHandle2 = 0
  set RSHandle3 = 0
  set QDHandle1 = 0
  set QDHandle2 = 0
  set WhichOne  = 0
  set ls1 = 0
  set ls2 = 0
end

-------------------------------------------------------------------------------
```

```
--InitDB
--
-- This function will initially open the
-- database and create a recordset that
-- will allow us to read and edit all of
-- the contacts in our database
-----------------------------------------------------------------------

on InitDB
  global dbHandle
  global rsHandle1
  global rsHandle2
  --
  --Open the contacts database assuming
  --that it exists in the local directory.
  --If it doesn't we will get an error back
  --that we can use to communicate with the
  --user.
  --
  set dbHandle = DGOpenDatabase(the moviePath & "kuronthogai.mdb")
  if dbHandle = "#ERROR#" then
    alert "Error: " & GetLastDGError()
    alert "Now exiting the program."
    ClearDGError()
    quit
  end if

-----------------------------------------------------------------------
  --This SQL string will define our
  --recordset. It essentially means
  --Select everything that exists
  --inside the Contacts table.
  --
  set sqlString1 = "SELECT * FROM tblMain"
  set sqlString2 = "SELECT * FROM tblList"
  --
  --Open a recordset using our above created
  --SQL string and our database handle
  --
  set rsHandle1 = DGCreateRecordset(sqlString1, dbHandle)
  set rsHandle2 = DGCreateRecordset(sqlString2, dbHandle)

  if rsHandle1 = "#ERROR#" or rsHandle2 = "#ERROR#" then
    alert "Error: " & GetLastDGError()
    alert "Now exiting the program."
    quit
```

```
  end if
end

--------------------------------------------------------------------------------
--CheckForDataGrip
--
-- Description: Will run through each of the Xtra
--              names and check to see if the
--              DataGrip Xtra is loaded.
--
--------------------------------------------------------------------------------

on CheckForDataGrip
  repeat with counter = 1 to (the number of xtras)
    set tmpText = tmpText && the name of xtra counter
  end repeat

  if not (tmpText contains "DAO_Xtra") then
    alert "Are you sure you have the DataGrip Xtra installed in the Xtras directory?"
  end if
end

--------------------------------------------------------------------------------
--Interface Scripts 1
--------------------------------
--FillFields
--
-- This functions will fill out the text
-- in the fields
--
--------------------------------------------------------------------------------

on FillFields
  global qdHandle
  global rsHandle1
  --
  --Check to see if we have any records
  --
  if DGRSGetRecordCount(rsHandle1) <= 0 then return
  --
  --Fill out these fields
  --
  SetFieldVal("FldSongNo", "º¹¥Ö ‡¯")
  SetFieldVal("FldFirstLine", "Ã°Ö ×Í")
  SetFieldVal("FldTitle", "º□Ò¿½")
  SetFieldVal("FldPoet", "½Ò×Ñ")
```

```
  SetFieldVal("FldLand","±□«")
  SetFieldVal("FldWho","˜íì")
  SetFieldVal("FldToWho","Æ'Í¥Å")
end

--------------------------------------------------------------------------------
--Interface Scripts 1
--------------------------------
--FillFields
--
-- This functions will fill out the text
-- in the fields
--
--------------------------------------------------------------------------------

on SetList dbField1, dbField2
  global rs1Handle

  set val= DGRSGetFieldValue(dbField2,rs1Handle)
  if val = "#ERROR#" then
    alert "Error" & RETURN & GetLastDGError()
    ClearDGError()
  else
    --
    --Run through the new recordset and
    --pull out all of the song numbers
    --
    repeat while not DGRSIsEOF(rs1Handle)
      set desc= DGRSGetFieldValue(dbField2,rs1Handle)
      if desc = "#ERROR#" then set desc = ""
      set num = DGRSGetFieldValue(dbField1,rs1Handle)
      if num = "#ERROR#" then set num = ""
      set tmpStr2 =  tmpStr2 & desc & "     " & num & RETURN
      DGRSMoveNext(rs1Handle)
    end repeat
    DGRSMoveFirst(rs1Handle)
    set the text of member 10 = string(tmpStr2)
  end if
end

--------------------------------------------------------------------------------
--DB Scripts
--------------------------------
--SetFieldVal
--
-- params:
```

```
-- fieldName: the name of the field on the stage
-- dbField : the field in the DB
--
-- Returns:
-- The value in the field. If an error
-- occurs nothing will be returned but
-- an error dialog will be displayed
--
-- Notes:
-- This function uses a global variable as
-- the recordset handle.
-----------------------------------------------------------------------------
on SetFieldVal fieldName, dbField
 global rsHandle1

 set val = DGRSGetFieldValue(dbFIeld, rsHandle1)
 if val = "#ERROR#" then
  alert "Error while retrieving from " & fieldName & RETURN & GetLastDGError()
  ClearDGError()
  set the text of member fieldName = ""
 else
  set the text of member fieldName = string(val)
 end if
end

-----------------------------------------------------------------------------
--ScrollField Scripts
-------------------------------
--FillScrollFileld1
--
-- This functions will query the
-- database for the information
-- and then fill out the scroll text
-- in the scrollfield1
-----------------------------------------------------------------------------

on FillScrollFileld1 dbfield1,dbfield2,query,identifier
 global rsHandle1
 global dbHandle
 global List
 global SongCounter

 set SongCounter = 0
 set List = []
 --
```

```
--Open a stored query that will retrieve all of
--the dbfield1 adn dbfield2
--
set qdHandle1 = DGQDOpen(query, dbHandle)
if qdHandle1 = "#ERROR#" then
  alert "Error: " & GetLastDGError()
  alert "Now exiting the program."
  quit
end if
--
--Check to see if we have any records
--
if DGRSGetRecordCount(rsHandle1) <= 0 then return
--
--Fill out the scroll fields
--
--
set val = DGRSGetFieldValue(dbfield1, rsHandle1)
if val <> "#ERROR#" then
  DGQDSetParameterValue("fldID",identifier, qdHandle1)
  set tmpRS = DGQDCreateRS(qdHandle1)
  --
  --Run through the new recordset and
  --pull out all of the song numbers
  --
  repeat while not DGRSIsEOF(tmpRS)
    set Var1 = DGRSGetFieldValue(dbfield1, tmpRS)
    if Var1 = "#ERROR#" then set Var1 = ""

    add List,Var1
    set Var2 = DGRSGetFieldValue(dbfield2, tmpRS)

    if Var2 = "#ERROR#" then set Var2 = ""

    set tmpStr1 = tmpStr1 & padText(Var1) & var2 &  "     " & RETURN
    set SongCounter = count(List)
    DGRSMoveNext(tmpRS)
  end repeat
  DGRSClose(tmpRS)
 end if
 set the text of member "ScrollList1" = string(tmpStr1)
end

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

-- PadText
--
```

```
-- Params;
--   str:
--    A string that needs to be padded
--
-- Returns:
--   A new string
--
-- Description:
--   that is padded to the appropriate
--   length so that it will display
--   correctly in a 2 column list box
--
-------------------------------------------------------------------------------

on PadText str
  --
  --Set the num of chars before the next column
  --
  set padLength = 20

  set count = length(str)

  if count <= (padLength - 1) then
    repeat with i = 1 to (padLength - count)
      set str = str & " "
    end repeat
  else
    set str = chars(str, 1, padLength - 1) & " "
  end if

  return str
end

-------------------------------------------------------------------------------
--ScrollField Scripts
-------------------------------
--FillScrollFileld2
--
-- This functions will query the
-- database for the information
-- and then fill out the scroll text
-- in the scrollfield1
-------------------------------------------------------------------------------

on FillScrollFileld2 dbfield
  global rsHandle2
```

```
  global dbHandle

  repeat while DGRSGetFieldValue(dbfield,rsHandle2)<>""
   set str = DGRSGetFieldValue(dbfield,rsHandle2)
   if str = "#ERROR#" then set Var1 = ""

   set tmpStr1 = tmpStr1 & str & RETURN
   DGRSMoveNext(rsHandle2)
  end repeat

  DGRSMoveFirst(rsHandle2)
  set the text of member "ScrollList2" = string(tmpStr1)
end

-------------------------------------------------------------------------------
--ScrollField Scripts
-------------------------------
--FillScrollFileld3
--
-- This functions will query the
-- database for the information
-- and then fill out the scroll text
-- in the scrollfield1
-------------------------------------------------------------------------------

on FillScrollFileld3 index, txtfield, dbfield
  global rsHandle1
  global dbHandle

  DGRSMove(index - 1, rsHandle1)
  SetFieldVal(txtfield,dbfield)
  DGRSMoveFirst(rsHandle1)
end

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

on FillList
  global List

  set total = count(List)
  set position = 1
  sort(List)
  repeat while position <= total

   set tmpStr1 = tmpStr1 & string(getAt(List, position)) & RETURN
   set position = position + 1
```

```
  end repeat

  set the text of member "FldScrollList3" = string(tmpStr1)
end

----------

on SetPicture
  set WhichPic = value(SongNo)
  erase member 1 of castlib "image"
  duplicate member member WhichPic of castlib "image", member 1 of castlib "image"
  set the name of member  1 of castlib "image" to "1"

  set picHeight = the height of member 1 of castlib "image"
  set picWidth  = the width  of member 1 of castlib "image"

  if (picHeight - picWidth * 3/4) > 0 then
    set the height of sprite 81 to 700
    set the width of sprite 81 to 450
  else
    set the height of sprite 81 to 650
    set the width of sprite 81 to 500
  end if
   set the memberNum of sprite 81 to 1
  updatestage
end

----------
```

USER MENUAL

MAIN PAGE

1. Page Title

Buttons:

2. Go to Page 2 (search according to First Line, Song Title, Song Number, Poet and Land)
3. Go to Page 3 (search according to Matter (*Karupporul)*)
4. Go to Page 4 (search according to Time (*Muthalporul)*)
5. Go to Page 5 (search according to Love(*Uripporul)*)

PAGE 2: SEARCH(First Line, Song Title, Song Number, Poet and Land)

1. Page Title

Buttons:

2. Search according to First Line
3. Search according to Song Title
4. Search according to Song Number
5. Search according to Poet
6. Search according to Land

Labels:

7. Label for scroll List 1 (Poet or Land)
8. Label for scroll List 2 (Song Number)
9. Label for scroll List 2 (First Line or Song Title)
10. Label (Total)
11. Label for Total Song

Scroll List:

12. Scroll List 2 views Song Numbers and First Lines
13. Scroll List 1 views Poet or Land

PAGE 3: MATTER

1. Page Title

Buttons:

2. Head
3. Deities
4. Food
5. Fauna
6. Flora
7. Bird
8. Drum
9. Occupation
10. Lyre
11. Water
12. Village

Labels:

13. Label for scroll List 1
14. Label for scroll List 2 (First Line)
15. Label for scroll List 2 (Song Number)
16. Label (Total)
17. Label for Total Song

Scroll List:

18. Scroll List 1 views Matter
19. Scroll List 2 views Song Numbers and First Lines

PAGE 4: TIME

1. Page Title

Buttons:

2. Division of day
3. Division of year

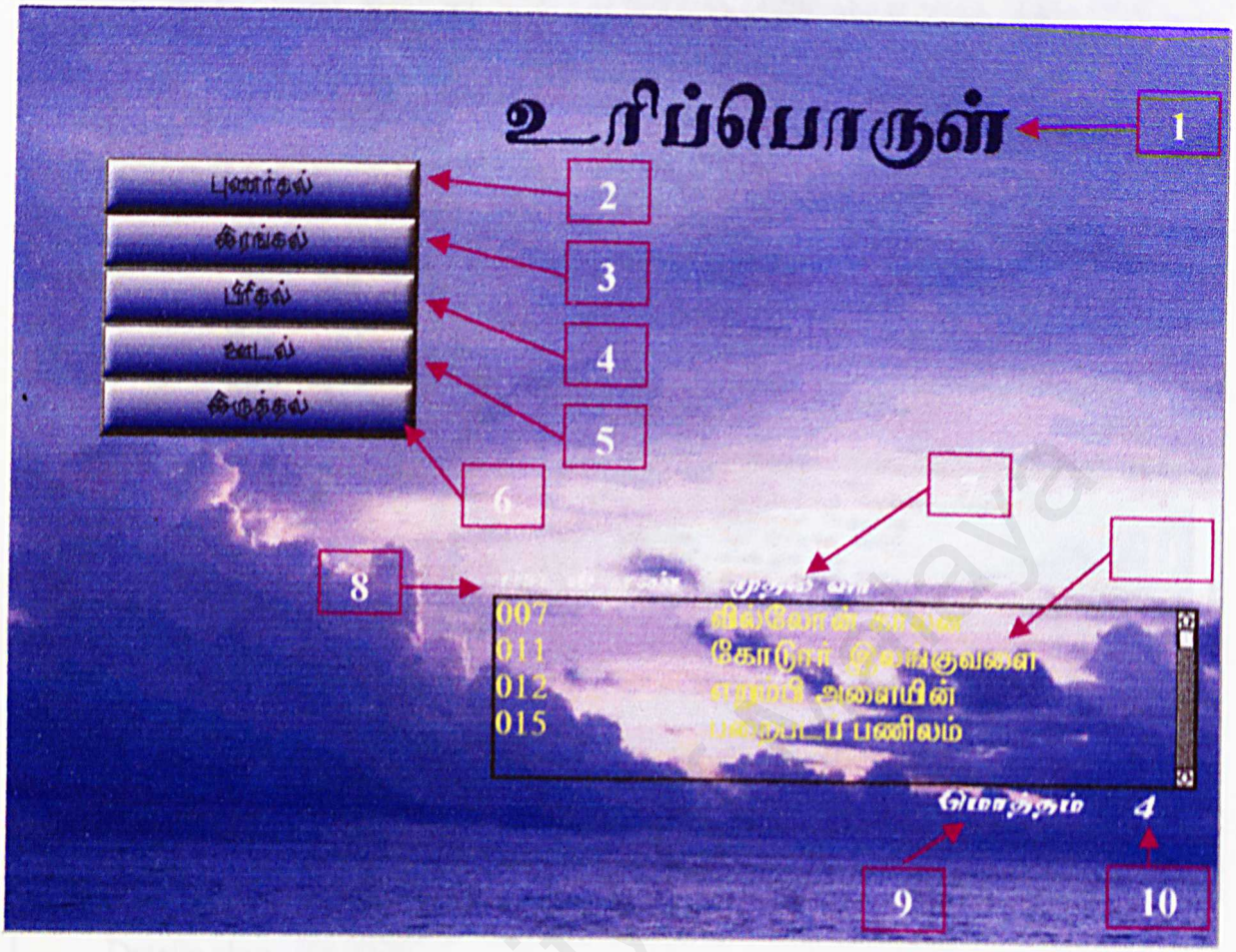
Label:

4. Label for scroll List 2 (First Line)
5. Label for scroll List 2 (Song Number)
6. Label (Total)
7. Label for Total Song

Scroll List:

8. Scroll List 1 views Division of day
9. Scroll List 2 views Song Numbers and First Lines

PAGE 5: LOVE



1. Page Title

Buttons:

2. Union
3. Bewailing
4. Separation
5. Sulking
6. Patience in Separation

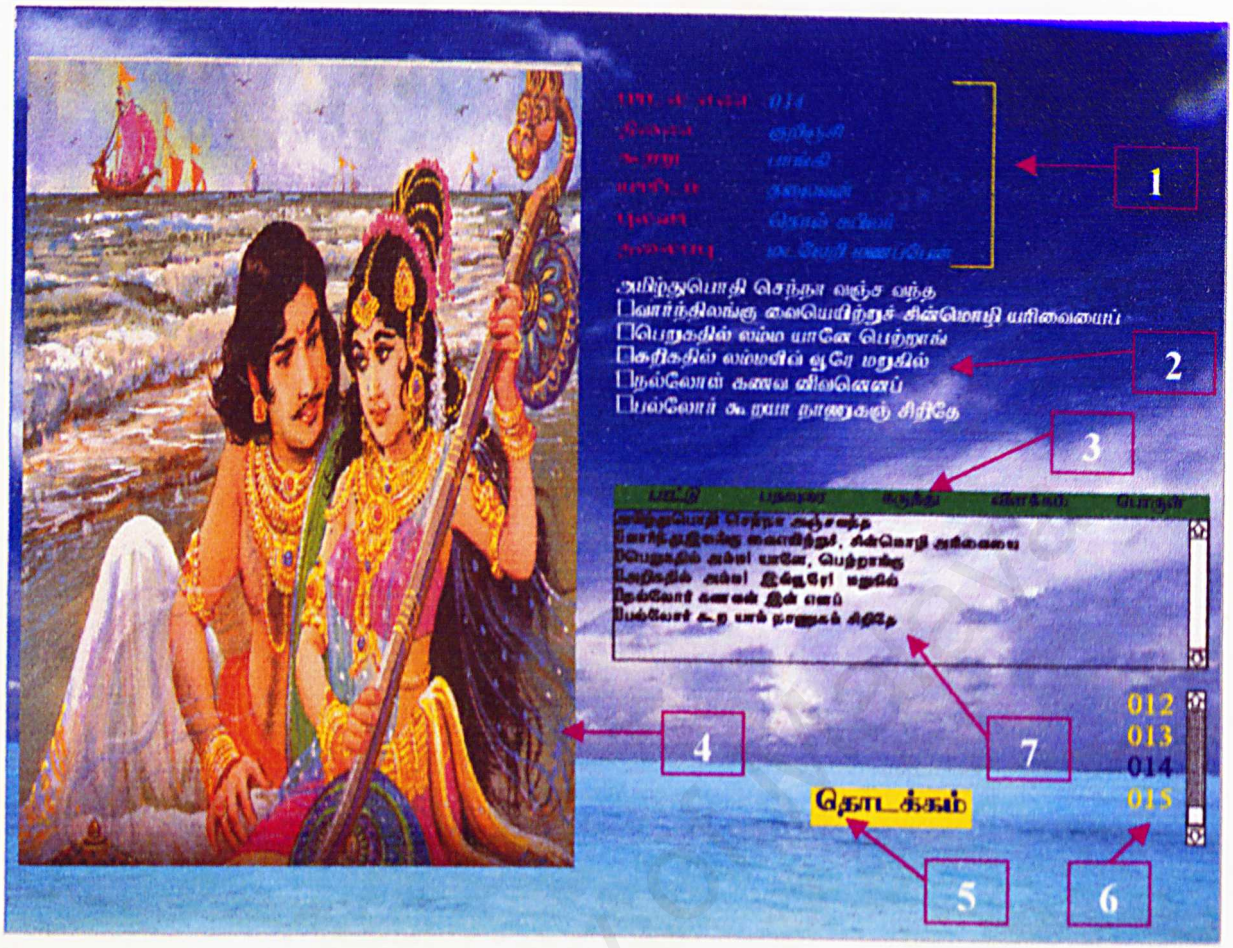
Label:

7. Label for scroll List 2 (First Line)
8. Label for scroll List 2 (Song Number)
9. Label (Total)
10. Label for Total Song

Scroll List:

10. Scroll List 2 views Song Numbers and First Lines

PAGE 6 : VIEW



1. Details about the poem
2. Lyrics of the poem
3. Buttons
4. Image
5. Button to Main Page
6. Scroll List with Song Numbers
7. Scroll List with explanations