# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

# JNANA SANGAMA, BELAGAVI-590018

#### 

# *A Mini Project Report on*

“Movie Database System”

Submitted in the partial fulfillment for the requirements for the conferment of degree of

BACHELOR OF ENGINEERING

In

#### **COMPUTER SCIENCE AND ENGINEERING**

#### Submitted By

**Mr. Pratyush Raj Shukla** USN: 1BY20CS139

**Mr. Piyush Parganiha** USN: 1BY20CS134

Under the guidance of

**Mrs. Ambika G.N**

Assistant Professor

Department of CSE, BMSIT&M.



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

# BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT

**(An Autonomous Institute, Affiliated to VTU, Belagavi**

**Avalahalli, Yelahanka, Bengaluru-560064)**

**2022-2023**

**BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**(An Autonomous Institute, Affiliated to VTU, Belagavi**

**Avalahalli, Yelahanka, Bengaluru-560064)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**CERTIFICATE**

This is to certify that the Mini Project work entitled **“Movie Database System”** is a bonafide work has been carried out by **Mr. Pratyush Raj Shukla (1BY20CS139) and Ms. Peeysuh Parganiha (1BY20CS144), bonafide movies of BMS Institute of Technology and Management, Autonomous Institute Affiliated to VTU,** in partial fulfillment for the award of **Bachelor of Engineering Degree in Department of Computer Science and Engineering** during the year 2022-23. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in this report. The Mini project report has been approved as it satisfies the academic requirements in respect of Mini project work for the B.EDegree.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Signature of the Guide1 Signature of the Guide2 Signature of the HOD**

**Mrs. Ambika G.N** **Mr. Rajesh N.V** **Dr. Thippeswamy .G**

Assistant Professor Assistant Professor Professor & HOD

Dept. Of CSE Dept. Of CSE Dept. Of CSE

BMSIT&M BMSIT&M BMSIT&M

**Name of the Examiners Signature with Date**

**1.**

**2.**

**ABSTRACT**

The aim of the project is to develop a system capable of doing CRUD operations, that is, Create, Read, Update and Delete the information of a particular movie from the database.

The four CRUD functions can perform different types of operations on selected data within the database. CRUD operations are widely used in many applications that are supported by underlying relational databases. The CRUD acronym identifies all of the major functions that are inherent to relational databases and the applications used to manage them, which include Oracle Database, Microsoft SQL Server, MySQL, and others.

A relational database consists of tables with rows and columns. In a relational database, each row of a table is known as a tuple or a record. Each column of the table represents a specific attribute or field. The four CRUD functions can be called by the users to perform different types of operations on selected data within the database. This could be accomplished using code or through a graphical user interface.

CRUD operations are also used to manage forums, eCommerce stores, social media websites and many, many other types of applications that are supported by a relational database.

Tasks of my project are development of an attractive and easy-to-use Graphical User Interface using python and its libraries and MySQL workbench for its database.

**ACKNOWLEDGEMENT**

We are happy to present this Mini project after completing it successfully. This project would not have been possible without the guidance, assistance and suggestions of many individuals. I would like to express my deep sense of gratitude and indebtedness to each and every one who has helped me to make this project a success.

We heartily thank our **Principal, Dr. MOHAN BABU G N,** **BMS Institute of Technology & Management, Autonomous Institute Affiliated to VTU** for his constant encouragement and inspiration in taking up this Mini project.

We heartily thank our **Head of the Department, Dr. Thippeswamy. G, Dept. of Computer Science** **and** **Engineering, BMS Institute of Technology & Management, Autonomous Institute Affiliated to VTU** for his constant encouragement and inspiration in taking up this Mini project.

We gracefully thank our Project guide**, Mrs. Ambika G.N, Assistant Professor, Dept. of** **Computer Science and Engineering,** for her encouragement and advice throughout the course of the Mini Project work.

Special thanks to all the staff members of Computer Science Department for their help and kind co-operation.

Lastly, we thank our parents and friends for their encouragement and support given to me in order to finish this Mini Project work. By,

**Pratyush Raj Shukla(1BY20CS139)**

**Peeysuh Parganiha(1BY20CS144)**

**CONTENTS**

**Chapters**

Chapter 1: **Introduction**

* 1. Brief Introduction
  2. Motivation ans Scope
  3. Objective
  4. Background and Related Work
  5. Uses of Movie Database System.
  6. Problem statement

Chapter 2: **Literature Survey**

Chapter 3: **Software Requirements Specification**

3.1 Hardware requirements

3.2 Modules of the system

3.3 Hardware interfaces

3.4 Software interfaces

3.5 Communication interfaces

3.6 User Documentations

3.7 Software requirements

3.8 Functional requirements

3.9 Non-Functional requirements

Chapter 4: **Design**

4.1 Schema Diagram

4.2 Entity-Relationship Diagram

Chapter 5: **Implementation**

5.1 Description of database used

5.2 MySQL server

5.3 MySQL workbench

5.4 Implementation with Screen shot

Chapter 6: **Conclusion & Future Enhancement**

6.1 Conclusion

6.2 Future Enhancement

Chapter 7: **References**

**CHAPTER 1**

**INTRODUCTION**

* 1. **BRIEF INTRODUCTION**

Movie Database System is a computerized system that allows users to store, manage, and retrieve information about movies. The system is typically designed to store a wide range of information about movies, including movie titles, release dates, cast and crew information, box office numbers, and ratings. This information can be easily accessed, searched, and analyzed by users, making it a useful tool for film enthusiasts, researchers, and industry professionals.

* 1. **Motivation and Scope**

The core component of a Movie Database System is a database management system (DBMS), which is responsible for storing, organizing, and managing the movie data. The DBMS can be a relational database, such as MySQL or PostgreSQL, or a NoSQL database, such as MongoDB or Cassandra. The database schema, which defines the structure of the data and the relationships between different pieces of information, is carefully designed to ensure that the data is organized in a logical and efficient manner.

1.3 **Objectives**

A Movie Database System can be used for a variety of purposes, such as movie research, film analysis, and entertainment. Users can query the database to retrieve specific information about movies, such as the top-grossing films of a certain year or the most popular movies in a specific genre. The system can also be used to generate reports and perform data analysis to identify trends in the film industry.

1.4 **Background and Related Work**

IMDB (Internet Movie Database) is one of the largest movie databases available in the web. It has detailed information about all the movies including movie show times and trailers. But it doesn’t provide the information about “online streaming”. Rotten Tomatoes is another website which primarily gives the combined critic score of any movie. It provides the basic movie data, streaming links but not show times of movies playing in theaters. Movie lens suggests the movies based on user’s ratings for other movies and basic data but no information on “show times” and “ online streaming”. Today, online streaming/renting a movie is widely popular and most of the movies are coming up with an option to rent the movie online at the same time when they are playing in theaters. PMDS covers the gap discussed above.

1.5 **Uses of Movie Database System**

Movie database systems have a variety of uses, some of which include:

* Movie Recommendation: A movie recommendation system uses data from a movie database to suggest movies to users based on their viewing history and preferences.
* Content-based Filtering: A content-based filtering system uses data from a movie database to recommend movies to users based on the characteristics of the movies they have previously watched or rated.
* Sentiment Analysis: A sentiment analysis system uses data from a movie database to determine the overall sentiment or emotion of a movie, such as positive, negative, or neutral.
* Content Management: A movie database can be used to store and manage large amounts of information about movies, such as their title, release date, cast, and crew.
* Search Functionality: A movie database can be used to provide search functionality that allows users to easily find movies based on various criteria, such as title, release date, genre, and cast.
* Data Visualization: A movie database can be used to create visualizations of movie data, such as box office earnings over time, trends in movie genres, or the most successful movie studios.

1.5 **Problem statement**

Designing and implementing a GUI based system to view the data about movie like year of release, director, production house and most importantly the Title and Language. User can find the movie details based on name, genre, actor name, year of release. The registered users can also rate the movies which will affect the overall user rating of the corresponding movie.

**Solution**

The system includes a user interface that allows users to interact with the database and perform tasks such as searching for movies, adding new movies, and updating existing movie information. The interface is usually web-based, allowing users to access the system from any device with internet access. The interface should be designed to be user-friendly and easy to navigate, making it accessible to users of all skill levels.

**CHAPTER 2**

**LITRETURE SURVEY**

In recent years, web application frameworks have been widely practiced by many developers to increase programming productivity as the framework are more flexible, rapidly built using CRUD operation, MVC-based, secure and most of them are published under an open-source license which will reduce the final cost of development. Although the CRUD automation in the web application framework boosts the development process, there are many important aspects of a web application absent from the CRUD output. Therefore, this multivocal literature review investigates the records management aspects that are required in modern WA and the perceived benefit of integrating the records management aspect into CRUD operation. The study extracted 284 publications from respectable scientific resources and the grey resources literature created by WA development practitioners outside academic mediums. After a detailed review process, only 14 scientific primary studies and 13 grey studies were considered for this review based on defined inclusion and exclusion criteria. The review shows that the most important aspect required in WA is search, role-based access control, retention, appraisal, search, audit trail, digital archiving, sharing, reporting, inactive files management and several other features. This important aspect has been analyzed and characterized according to its function and features. The method and procedure for integrating the specified aspect into CRUD operation are identified and discussed. Integrating and implementing the specified record management features into CRUD operation will boost the WA development productivity by producing more features as a standard output with integrated records management functions.

**CHAPTER 3**

**Software Requirements Specification**

3.1 **Hardware Requirements**

* Processor: Intel Pentium 4 or more.
* Ram: 1 GB or more, and database memory.
* Hard disk: 40 GB hard disk recommended for the primary partition.

3.2 **Modules of Movie Database System**

The system has a different segment to process a specific task which is the modules. This will help the system to be developed easily and make it more user friendly.

* **Registration module:**
* In this module, the movie will get registered as it is new in the Database. It will be formed like a structure where all the movie details will be filled.
* As this module is present online, the movie can register them from anywhere on the internet is present. After registration information will go to the admin for authentication.
* This module will reduce the hectic task of taking multiple forms from the institute and filling them carefully as any mistake will lead to getting new sets of the forms.
* **Search module:**

In the database there will be thousands of movies and suppose from this there is a need to find the detail of specific movies. The only information provided to search is the name of the movie. In the manual system, it will be catastrophic to find the movie as it is a very tedious job to do so. But with the computerized system admin can easily find the specific movie by just typing the name and click the search button. This module will help the admin in searching the movie record for alteration and maintenance.

3.3 **Hardware Interfaces**

* We require LAN connection for interacting with the database.
* local computers for any help or any other requirement.
* We use TCP/IP protocol for communicating with local hosts.

3.4 **Software Interfaces**

* We use Python programming language for writing the code for the project.
* Using GUI for interacting with the database.
* SQL server is used for creating the local and global database (server).
* Operating system: Windows XP or higher version.

3.5 **Communication Interfaces**

* The communication functions required by this product are LAN connection within the whole company so that anyone can interact with each other.
* We use TCP/IP protocol.

3.6 **User Documentation**

In our user manual, we are going to keep the information regarding our product which can be understandable by a new person who is going to use it. If a new person is using it online help will be provided in that we are going to explain each and every step clearly why our product can be useful for any user.

3.7 **Software Requirements**

The software required for the development of the project is:

Operation System: Windows 2000 Professional

Environment: Pycharm Community Edition

Framework: Version 1.0

Net Backend: SQL Server 2000

3.8 **Functional Requirement**

* Creation of the new record for the new movie.
* Deletion of the record which already exists in the system based on the requirement of the institute.
* Update in the record which is present in the system as per the need.

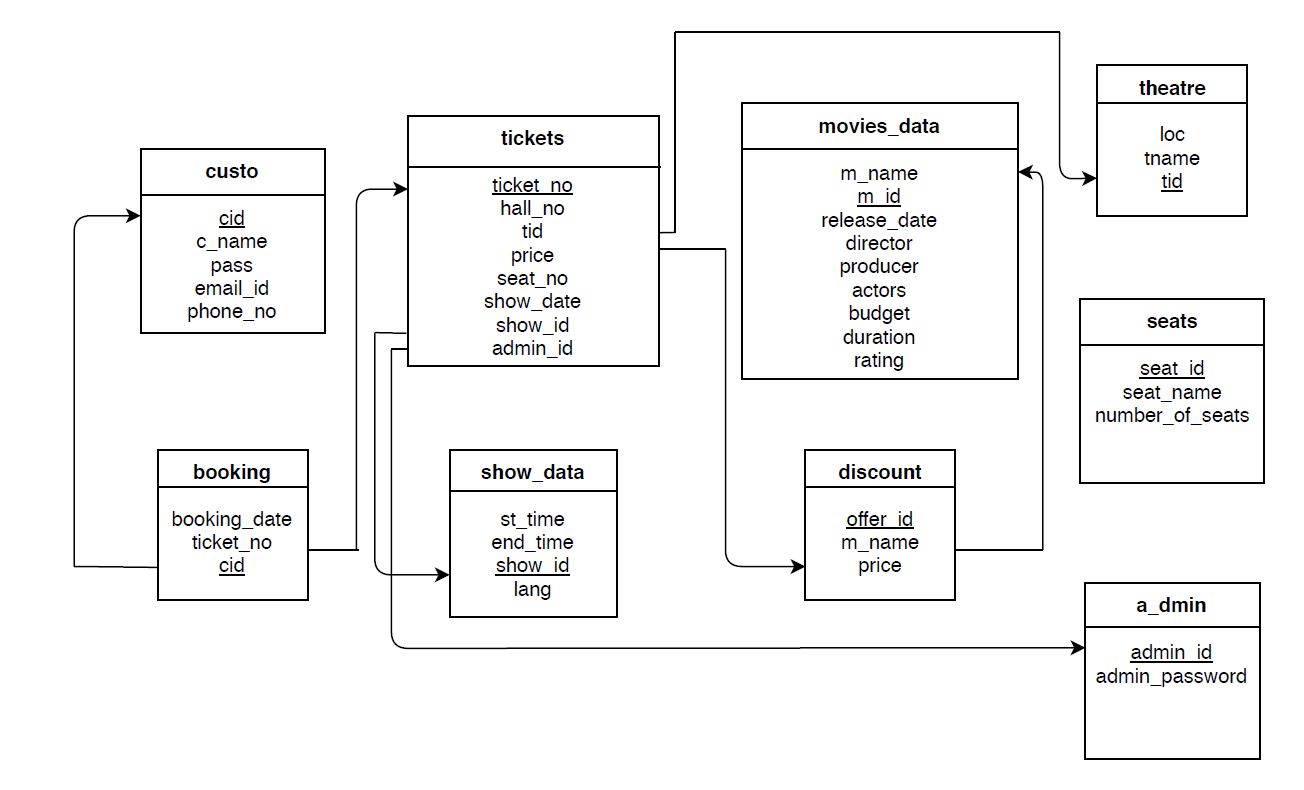
3.9 **Non-Functional Requirement**

* The security of the system is maintained by providing a login interface to the user. Only those who have the login is and password can enter the system.
* User-Friendly as the system is very interactive and can be easily operated.
* Maintainability and reliability if the system is kept very thoroughly as all the records kept in the database have the backups and system can restore if there is power loss.

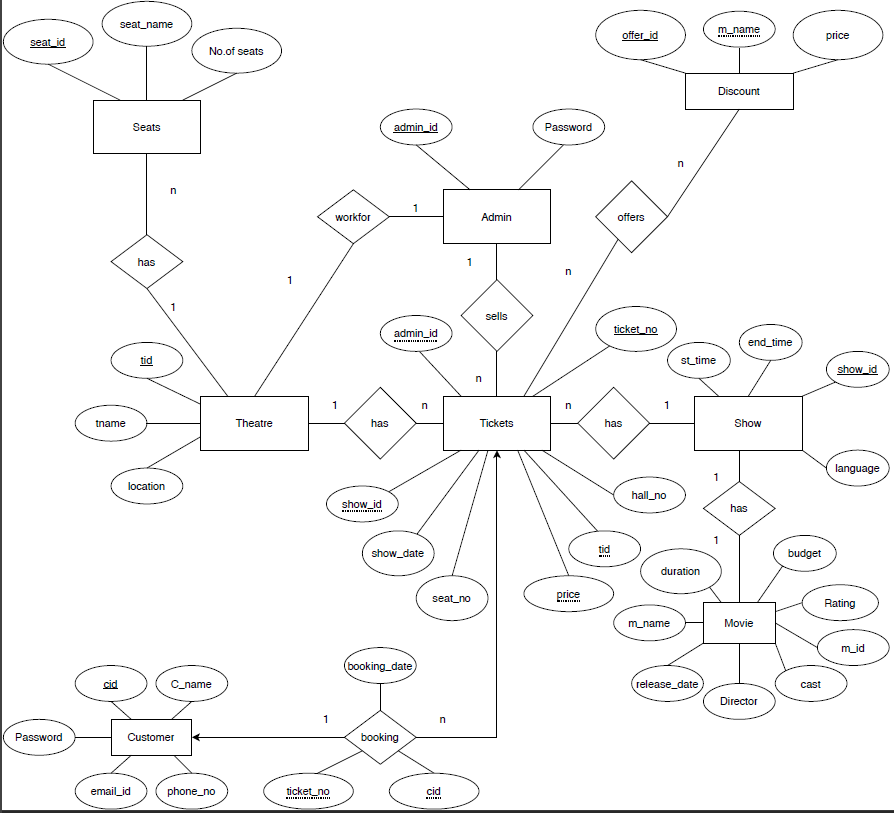
**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 SCHEMA DIAGRAM**



**4.2 Entity-Relationship Diagram**



**CHAPTER 5**

**IMPLEMENTATION**

**5.1 Description of Database used**

The database language used to create the DBMS used in the project is the SQL (Structured query language). It is a domain-specific language used in programming and designed for managing data hold in a Relational Database Management System (RDBMS), or for stream processing in a Relational Data Stream Management System (RDSMS). In comparison to older read/write APIs like ISAM or VSAM, SOL offers two main advantages: first, it introduced the concept of accessing many records with one single command; and second, it eliminates the need to specify how to reach a record, e.g with or without an index. Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SOL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements. SQL was one of the first commercial languages for Edgar F. Codd's relational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language. SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

5.2 **MySQL server**

The MySQL server provides a database management system with querying and connectivity capabilities, as well as the ability to have excellent data structure and integration with many different platforms. It can handle large databases reliably and quickly in high-demanding production environments. The MySQL server also provides rich function such as its connectivity, speed, and security that make it suitable for accessing databases.

The MySQL server works in a client and server system. This system includes a multiple-threaded SQL server that supports varied backends, different client programs and libraries, administrative tools, and many application programming interfaces (API)s.

5.3 **MySQL workbench**

MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more. MySQL Workbench is available on Windows, Linux and Mac OS X.

**Design:** MySQL Workbench enables a DBA, developer, or data architect to visually design, model, generate, and manage databases. It includes everything a data modeler needs for creating complex ER models, forward and reverse engineering, and also delivers key features for performing difficult change management and documentation tasks that normally require much time and effort.

**Develop:** MySQL Workbench delivers visual tools for creating, executing, and optimizing SQL queries. The SQL Editor provides color syntax highlighting, auto-complete, reuse of SQL snippets, and execution history of SQL. The Database Connections Panel enables developers to easily manage standard database

connections, including MySQL Fabric. The Object Browser provides instant access to database schema and objects.

**Administer:** MySQL Workbench provides a visual console to easily administer MySQL environments and gain better visibility into databases. Developers and DBAs can use the visual tools for configuring servers, administering users, performing backup and recovery, inspecting audit data, and viewing database health.

**Visual Performance Dashboard:** MySQL Workbench provides a suite of tools to improve the performance of MySQL applications. DBAs can quickly view key performance indicators using the Performance Dashboard. Performance Reports provide easy identification and access to IO hotspots, high cost SQL statements, and more. Plus, with 1 click, developers can see where to optimize their query with the improved and easy to use Visual Explain Plan.

**Database Migration:** MySQL Workbench now provides a complete, easy to use solution for migrating Microsoft SQL Server, Microsoft Access, Sybase ASE, PostreSQL, and other RDBMS tables, objects and data to MySQL. Developers and DBAs can quickly and easily convert existing applications to run on MySQL both on Windows and other platforms. Migration also supports migrating from earlier versions of MySQL to the latest releases.

5.4 **Implementation with Screenshots**

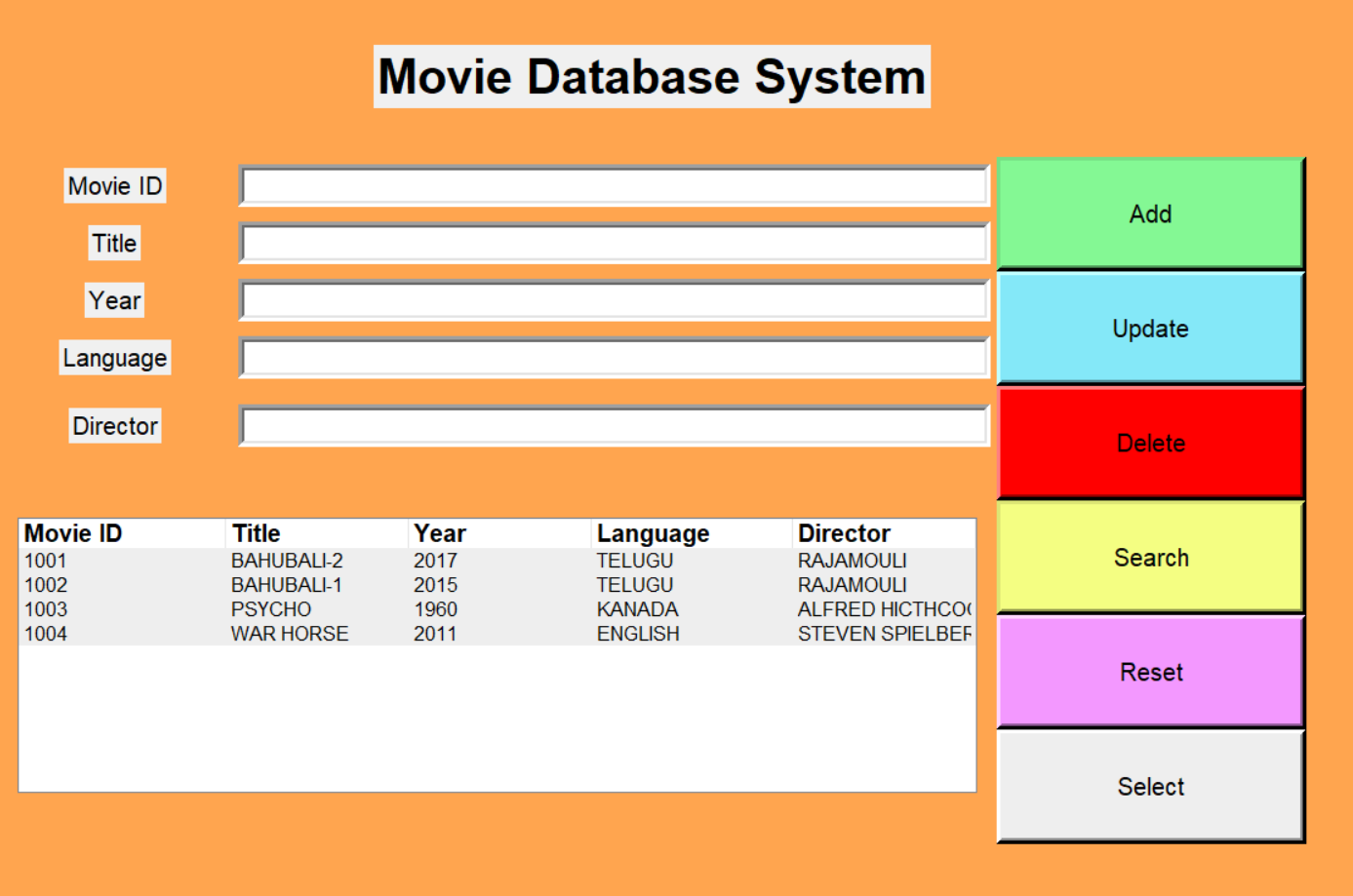


Fig 5.4a

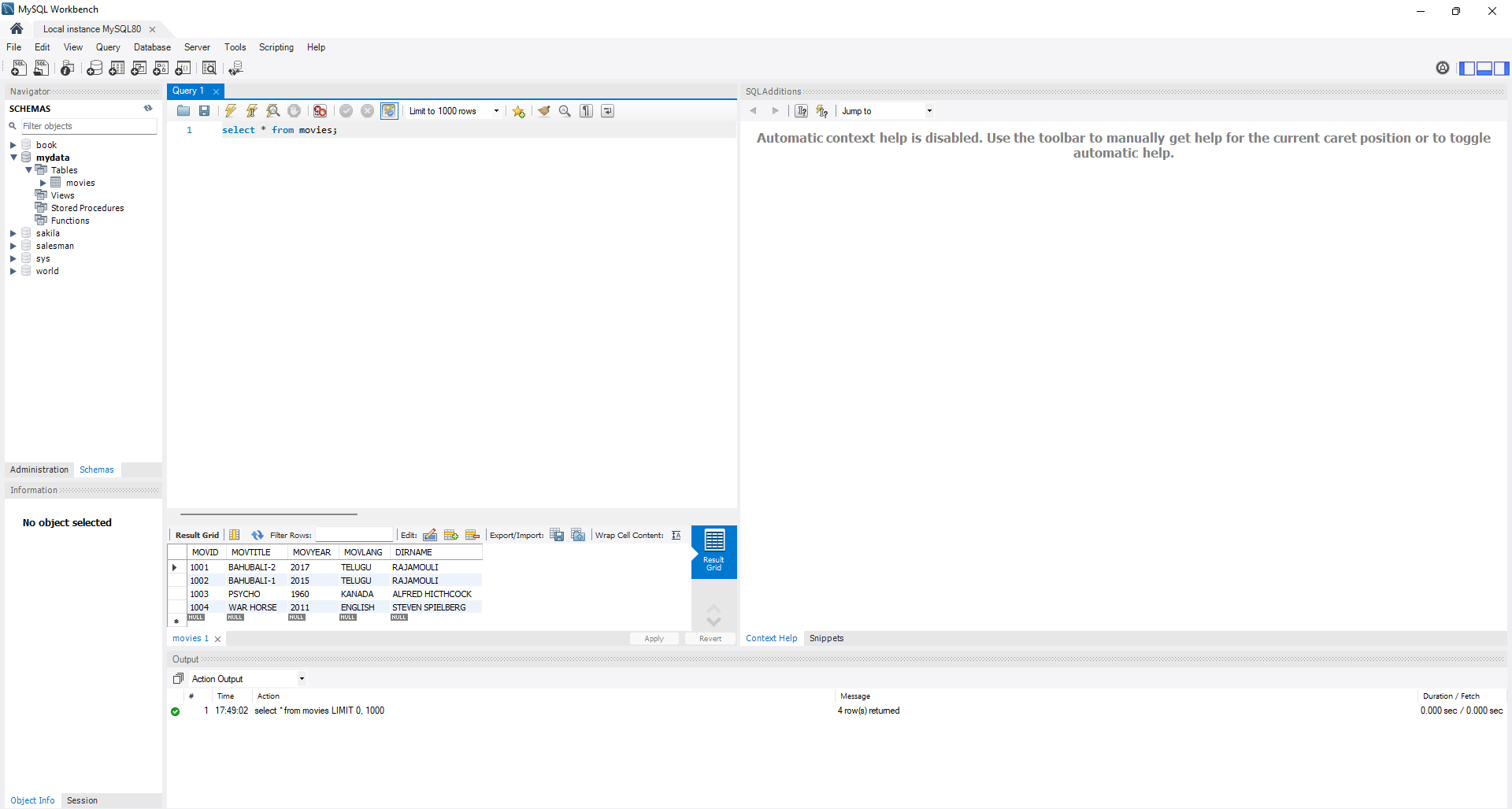


Fig 5.4b

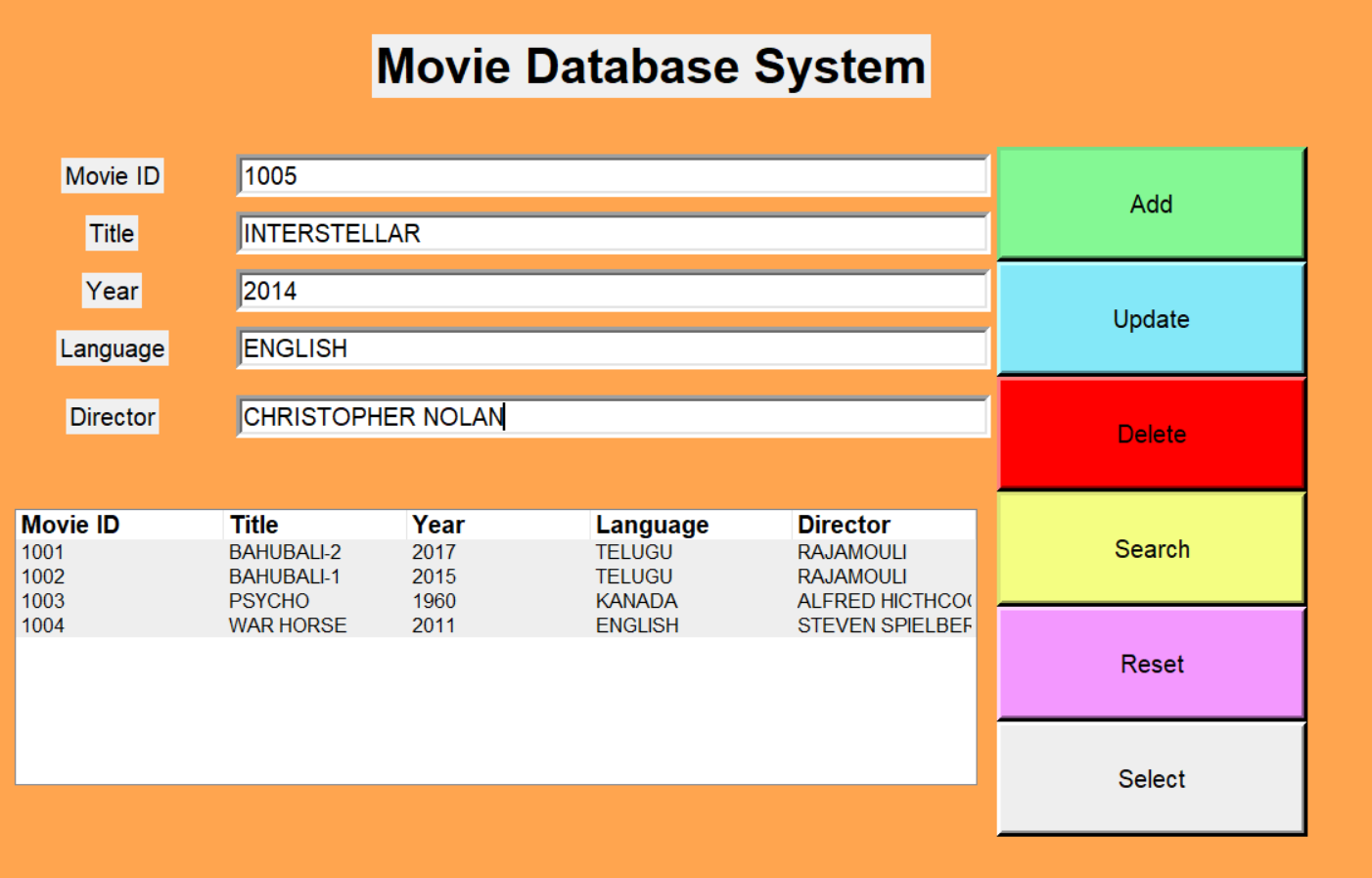


Fig 5.4 c.

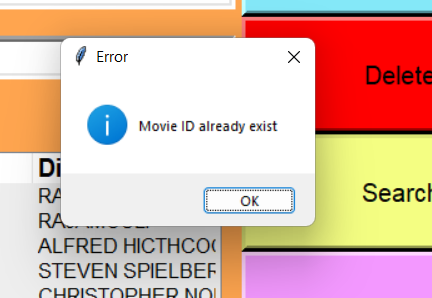


Fig 5.4d

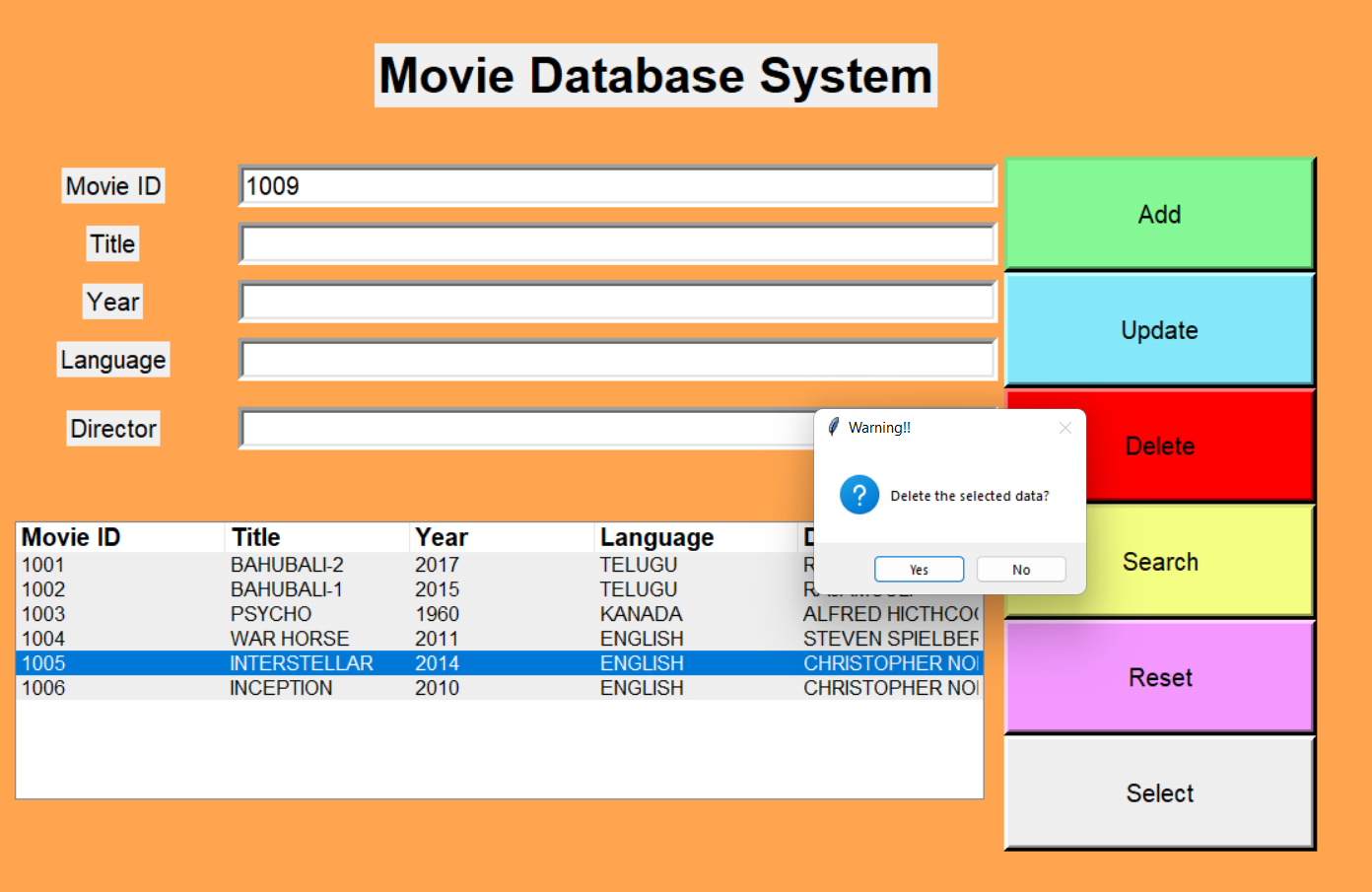


Fig 5.4e

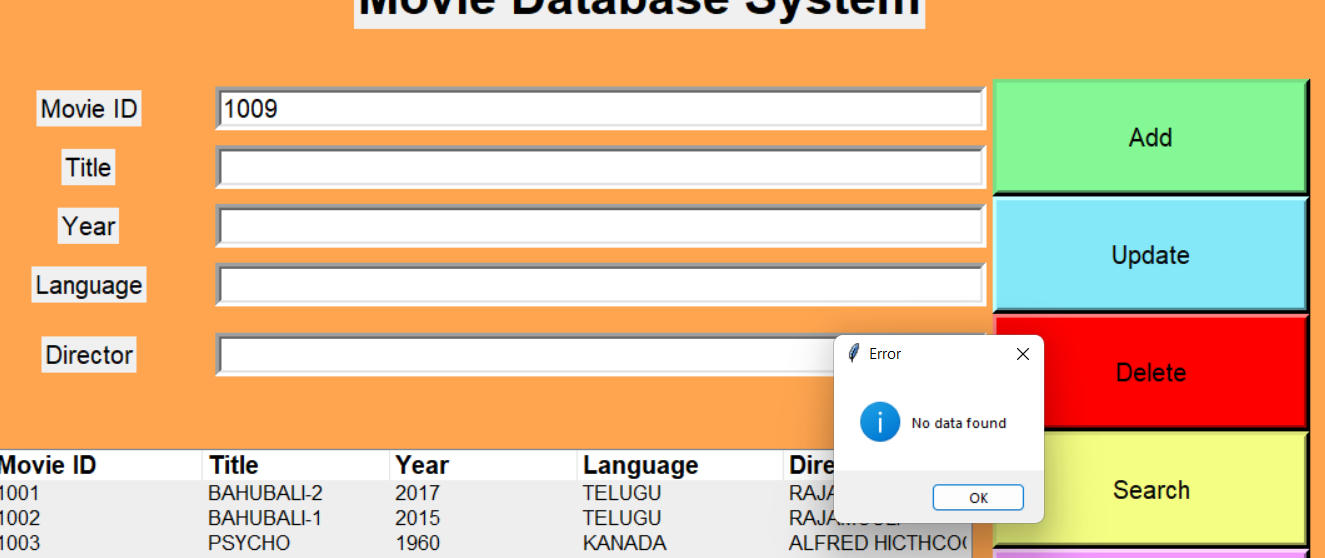


Fig 5.4f

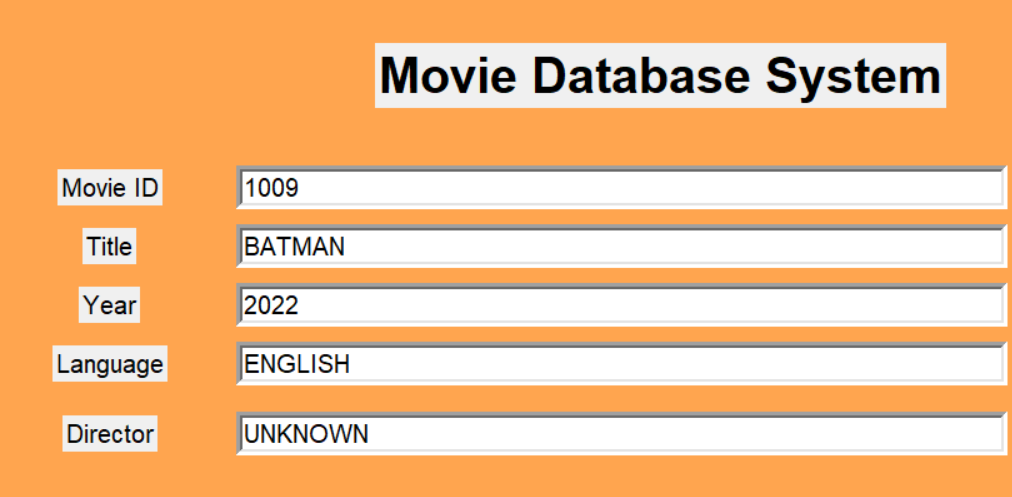


Fig 5.4g

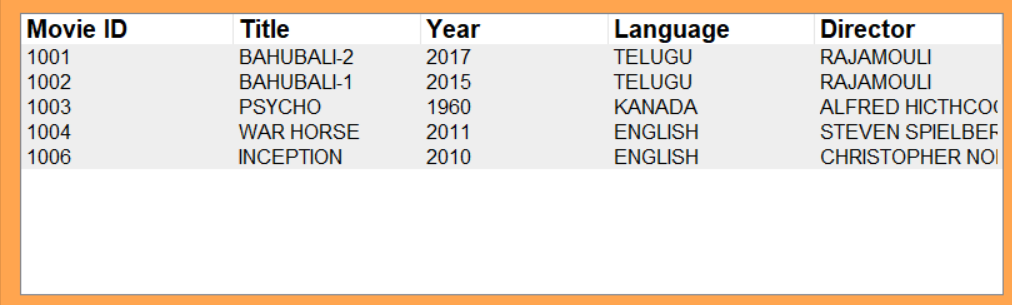


Fig 5.4h

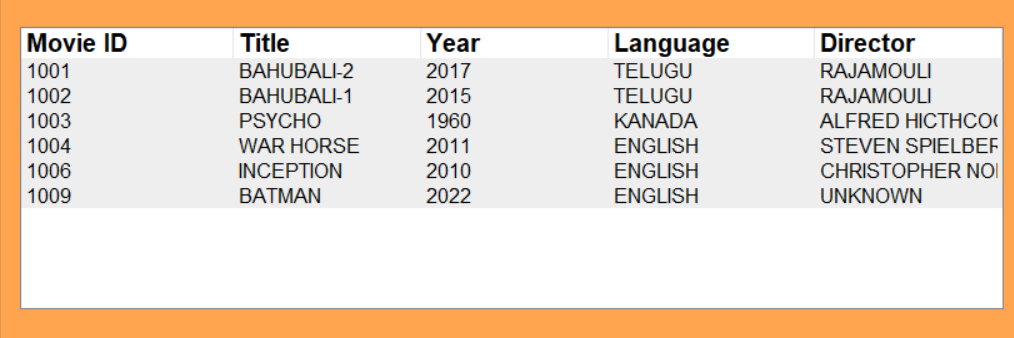


Fig 5.4i

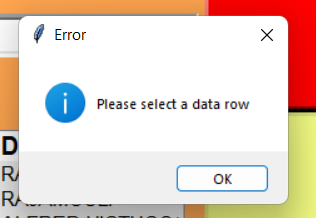


Fig 5.4j

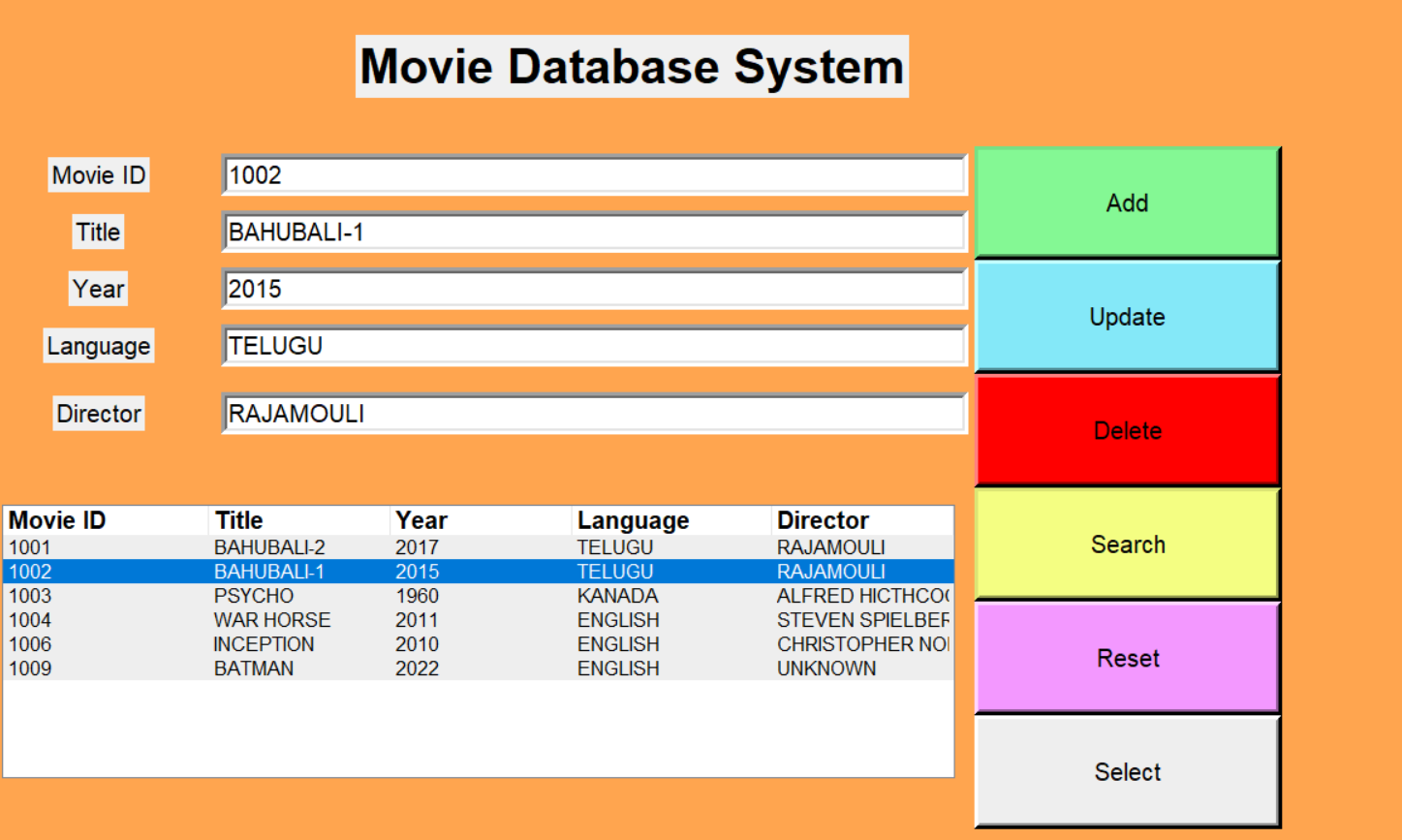


Fig 5.4k

**Code:**

**#pip install PyMySQL**

**import pymysql**

**from tkinter import \***

**from tkinter import ttk**

**from tkinter import messagebox**

**import tkinter as tk**

**# connection for phpmyadmin**

**def connection():**

**conn = pymysql.connect(**

**host='localhost',**

**user='root',**

**password='Pratyush23!!',**

**db='mydata',**

**)**

**return conn**

**def refreshTable():**

**for data in my\_tree.get\_children():**

**my\_tree.delete(data)**

**for array in read():**

**my\_tree.insert(parent='', index='end', iid=array, text="", values=(array), tag="orow")**

**my\_tree.tag\_configure('orow', background='#EEEEEE', font=('Arial', 12))**

**my\_tree.grid(row=8, column=0, columnspan=5, rowspan=11, padx=10, pady=20)**

**root = Tk()**

**root.title("Movie Database System")**

**root.geometry("1080x720")**

**root.configure(bg='green')**

**my\_tree = ttk.Treeview(root)**

**# placeholders for entry**

**ph1 = tk.StringVar()**

**ph2 = tk.StringVar()**

**ph3 = tk.StringVar()**

**ph4 = tk.StringVar()**

**ph5 = tk.StringVar()**

**# placeholder set value function**

**def setph(word, num):**

**if num == 1:**

**ph1.set(word)**

**if num == 2:**

**ph2.set(word)**

**if num == 3:**

**ph3.set(word)**

**if num == 4:**

**ph4.set(word)**

**if num == 5:**

**ph5.set(word)**

**def read():**

**conn = connection()**

**cursor = conn.cursor()**

**cursor.execute("SELECT \* fROM movies")**

**results = cursor.fetchall()**

**conn.commit()**

**conn.close()**

**return results**

**def add():**

**movid = str(movidEntry.get())**

**movtitle = str(movtitleEntry.get())**

**movyear = str(movyearEntry.get())**

**movlang = str(movlangEntry.get())**

**dirname = str(dirnameEntry.get())**

**if (movid == "" or movid == " ") or (movtitle == "" or movtitle == " ") or (movyear == "" or movyear == " ") or (**

**movlang == "" or movlang == " ") or (dirname == "" or dirname == " "):**

**messagebox.showinfo("Error", "Please fill up the blank entry")**

**return**

**else:**

**try:**

**conn = connection()**

**cursor = conn.cursor()**

**cursor.execute(**

**"INSERT INTO movies VALUES ('" + movid + "','" + movtitle + "','" + movyear + "','" + movlang + "','" + dirname + "') ")**

**conn.commit()**

**conn.close()**

**except:**

**messagebox.showinfo("Error", "Movie ID already exist")**

**refreshTable()**

**def reset():**

**decision = messagebox.askquestion("Warning!!", "Delete all data?")**

**if decision !="yes":**

**return**

**else:**

**try:**

**conn = connection()**

**cursor = conn.cursor()**

**cursor.execute("DELETE fROM movies")**

**conn.commit()**

**conn.close()**

**except:**

**messagebox.showinfo("Error", "Sorry an error occured")**

**return**

**refreshTable()**

**def delete():**

**decision = messagebox.askquestion("Warning!!", "Delete the selected data?")**

**if decision != "yes":**

**return**

**else:**

**selected\_item = my\_tree.selection()[0]**

**deleteData = str(my\_tree.item(selected\_item)['values'][0])**

**try:**

**conn = connection()**

**cursor = conn.cursor()**

**cursor.execute("DELETE fROM movies WHERE movid='" + str(deleteData) + "'")**

**conn.commit()**

**conn.close()**

**except:**

**messagebox.showinfo("Error", "Sorry an error occured")**

**return**

**refreshTable()**

**def select():**

**try:**

**selected\_item = my\_tree.selection()[0]**

**movid = str(my\_tree.item(selected\_item)['values'][0])**

**movtitle = str(my\_tree.item(selected\_item)['values'][1])**

**movyear = str(my\_tree.item(selected\_item)['values'][2])**

**movlang = str(my\_tree.item(selected\_item)['values'][3])**

**dirname = str(my\_tree.item(selected\_item)['values'][4])**

**setph(movid, 1)**

**setph(movtitle, 2)**

**setph(movyear, 3)**

**setph(movlang, 4)**

**setph(dirname, 5)**

**except:**

**messagebox.showinfo("Error", "Please select a data row")**

**def search():**

**movid = str(movidEntry.get())**

**movtitle = str(movtitleEntry.get())**

**movyear = str(movyearEntry.get())**

**movlang = str(movlangEntry.get())**

**dirname = str(dirnameEntry.get())**

**conn = connection()**

**cursor = conn.cursor()**

**cursor.execute("SELECT \* fROM movies WHERE movid='" +**

**movid + "' or movtitle='" +**

**movtitle + "' or movyear='" +**

**movyear + "' or movlang='" +**

**movlang + "' or dirname='" +**

**dirname + "' ")**

**try:**

**result = cursor.fetchall()**

**for num in range(0, 5):**

**setph(result[0][num], (num + 1))**

**conn.commit()**

**conn.close()**

**except:**

**messagebox.showinfo("Error", "No data found")**

**def update():**

**selectedmovid = ""**

**try:**

**selected\_item = my\_tree.selection()[0]**

**selectedmovid = str(my\_tree.item(selected\_item)['values'][0])**

**except:**

**messagebox.showinfo("Error", "Please select a data row")**

**movid = str(movidEntry.get())**

**movtitle = str(movtitleEntry.get())**

**movyear = str(movyearEntry.get())**

**movlang = str(movlangEntry.get())**

**dirname = str(dirnameEntry.get())**

**if (movid == "" or movid == " ") or (movtitle == "" or movtitle == " ") or (movyear == "" or movyear == " ") or (**

**movlang == "" or movlang == " ") or (dirname == "" or dirname == " "):**

**messagebox.showinfo("Error", "Please fill up the blank entry")**

**return**

**else:**

**try:**

**conn = connection()**

**cursor = conn.cursor()**

**cursor.execute("UPDATE movies SET movid='" +**

**movid + "', movtitle='" +**

**movtitle + "', movyear='" +**

**movyear + "', movlang='" +**

**movlang + "', dirname='" +**

**dirname + "' WHERE movid='" +**

**selectedmovid + "' ")**

**conn.commit()**

**conn.close()**

**except:**

**messagebox.showinfo("Error", "Movie ID already exist")**

**return**

**refreshTable()**

**label = Label(root, text="Movie Database System", font=('Arial Bold', 30))**

**label.grid(row=0, column=0, columnspan=8, rowspan=2, padx=50, pady=40)**

**movidLabel = Label(root, text="Movie ID", font=('Arial', 15))**

**movtitleLabel = Label(root, text="Title", font=('Arial', 15))**

**movyearLabel = Label(root, text="Year", font=('Arial', 15))**

**movlangLabel = Label(root, text="Language", font=('Arial', 15))**

**dirnameLabel = Label(root, text="Director", font=('Arial', 15))**

**movidLabel.grid(row=3, column=0, columnspan=1, padx=50, pady=5)**

**movtitleLabel.grid(row=4, column=0, columnspan=1, padx=50, pady=5)**

**movyearLabel.grid(row=5, column=0, columnspan=1, padx=50, pady=5)**

**movlangLabel.grid(row=6, column=0, columnspan=1, padx=50, pady=5)**

**dirnameLabel.grid(row=7, column=0, columnspan=1, padx=50, pady=5)**

**movidEntry = Entry(root, width=55, bd=5, font=('Arial', 15), textvariable=ph1)**

**movtitleEntry = Entry(root, width=55, bd=5, font=('Arial', 15), textvariable=ph2)**

**movyearEntry = Entry(root, width=55, bd=5, font=('Arial', 15), textvariable=ph3)**

**movlangEntry = Entry(root, width=55, bd=5, font=('Arial', 15), textvariable=ph4)**

**dirnameEntry = Entry(root, width=55, bd=5, font=('Arial', 15), textvariable=ph5)**

**movidEntry.grid(row=3, column=1, columnspan=4, padx=5, pady=0)**

**movtitleEntry.grid(row=4, column=1, columnspan=4, padx=5, pady=0)**

**movyearEntry.grid(row=5, column=1, columnspan=4, padx=5, pady=0)**

**movlangEntry.grid(row=6, column=1, columnspan=4, padx=5, pady=0)**

**dirnameEntry.grid(row=7, column=1, columnspan=4, padx=5, pady=0)**

**addBtn = Button(**

**root, text="Add", padx=65, pady=25, width=10,**

**bd=5, font=('Arial', 15), bg="#84f894", command=add)**

**updateBtn = Button(**

**root, text="Update", padx=65, pady=25, width=10,**

**bd=5, font=('Arial', 15), bg="#84E8f8", command=update)**

**deleteBtn = Button(**

**root, text="Delete", padx=65, pady=25, width=10,**

**bd=5, font=('Arial', 15), bg='red', command=delete)**

**searchBtn = Button(**

**root, text="Search", padx=65, pady=25, width=10,**

**bd=5, font=('Arial', 15), bg="#f4fE82", command=search)**

**resetBtn = Button(**

**root, text="Reset", padx=65, pady=25, width=10,**

**bd=5, font=('Arial', 15), bg="#f398ff", command=reset)**

**selectBtn = Button(**

**root, text="Select", padx=65, pady=25, width=10,**

**bd=5, font=('Arial', 15), bg="#EEEEEE", command=select)**

**addBtn.grid(row=3, column=5, columnspan=1, rowspan=2)**

**updateBtn.grid(row=5, column=5, columnspan=1, rowspan=2)**

**deleteBtn.grid(row=7, column=5, columnspan=1, rowspan=2)**

**searchBtn.grid(row=9, column=5, columnspan=1, rowspan=2)**

**resetBtn.grid(row=11, column=5, columnspan=1, rowspan=2)**

**selectBtn.grid(row=13, column=5, columnspan=1, rowspan=2)**

**style = ttk.Style()**

**style.configure("Treeview.Heading", font=('Arial Bold', 15))**

**my\_tree['columns'] = ("Movie ID", "Title", "Year", "Language", "Director")**

**my\_tree.column("#0", width=0, stretch=NO)**

**my\_tree.column("Movie ID", anchor=W, width=170)**

**my\_tree.column("Title", anchor=W, width=150)**

**my\_tree.column("Year", anchor=W, width=150)**

**my\_tree.column("Language", anchor=W, width=165)**

**my\_tree.column("Director", anchor=W, width=150)**

**my\_tree.heading("Movie ID", text="Movie ID", anchor=W)**

**my\_tree.heading("Title", text="Title", anchor=W)**

**my\_tree.heading("Year", text="Year", anchor=W)**

**my\_tree.heading("Language", text="Language", anchor=W)**

**my\_tree.heading("Director", text="Director", anchor=W)**

**refreshTable()**

**root.mainloop()**

**CHAPTER 6**

**Conclusion And Future Enhancement**

6.1 **Conclusion**

Overall, movie databases can be used for a wide range of applications, from simple data storage and retrieval to more complex systems that use machine learning and data mining techniques to provide personalized movie recommendations or to analyze patterns in movie data.

6.2 **Future Enhancements**

The project has a very vast scope in future. The project can be implemented on intranet in future. Project can be updated in near future as and when requirement for the same arises, as it is very flexible in terms of expansion. With the proposed software of database Space Manager ready and fully functional the client is now able to manage and hence run the entire work in a much better, accurate and error free manner. The following are the future scope for the project.

* Discontinue of particular Movie eliminate potential Ratings.
* Bar code Reader based Resigeration system.
* Individual Movie Detail system with photo using movie login

**CHAPTER 7**

**REFERENCES**

[1]. R . Ramakrishnan & J. Gehrke , "Database management systems," McGrawHill, Boston, MA, Third ed., 2003.

[2] .P. G. Selinger, M. Astrahan , D. Chamberlin, R. Lorie, and T. Price, "Access path selection in a relational database management system," in Proceedings of ACM-SIGMOD International Conference on Management of Data, pp. 22-34, Boston, June 1979.

[3]. Database systems Models, Languages, Design and Application Programming, RamezElmasri and Shamkant B. Navathe, 7th Edition, 2017

[4].Elamasri R . and Navathe, S., Fundamentals of Database Systems (3 rd Edition),

Pearson Education,2000.

[5].My$QSupport.http://www.mysql.com/whymysql/whitepapers/mysql \_wp\_enterprise\_

[6]. itSourcecode.com

[7]. Sites.google.com

[8]. Techvidan.com

[9]. GeeksForGeeks.com