SOFTWARE DESIGN SPECIFICATION

1.0 Introduction

An expert system is a computer system that is designed to emulate the decision-making ability of human experts. Expert systems are used in various domains, such as medical diagnosis, fault diagnosis, interpreting measurement data, and configuring systems.

The three main components of an expert system are a knowledge base, an inference engine, and a user interface. The knowledge base stores information about the particular domain, while the inference engine uses that information to solve problems. The user interface allows experts to input data and interact with the system.

Expert systems are successful in various domains due to their ability to improve the productivity of human experts, provide expertise uniformly and impartially, and save time. However, expert systems must be designed carefully, considering the particular domain and the types of reasoning used.

In rule-based expert systems, the knowledge base consists of a set of rules that the system uses to solve problems. Case-based reasoning systems, on the other hand, reason by analogy, using past cases to solve new problems.

Expert systems can be used in a variety of domains to improve the productivity of human experts. In rule-based expert systems, the knowledge base consists of a set of rules that the system uses to solve problems. Case-based reasoning systems, on the other hand, reason by analogy, using past cases to solve new problems.

1.1 Goals and objectives

1. The goal of developing a GUI application using Tkinter is to create a fast and easy way to create a swear analysis chatbot or tool. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit, which makes it easy to create the necessary widgets for the application.
2. The goal of adding widgets to the GUI application is to provide a means for inputting student information and analyzing performance. The devices added to the application will allow inputting student information and answering questions. The answers to the questions will be used to generate a grade for the student.
3. The goal of using the pack (), grid(), and place() geometry managers is to provide a means of organizing the widgets in the parent widget area. These geometry managers will allow the devices to be placed in a specific order and position.
4. The objective of providing grades to the students based on their answers is to give them feedback on their performance. The rates will be based on the answers to the questions asked in the application.

1.2 Statement of scope

The software is a system for analyzing student performance. It takes input from the student information form, verbal, and quantitative questions. It then processes these inputs to generate outputs. The outputs include the student's performance data, grades, and a report.

The software is designed to help teachers and administrators understand how well students are doing in school.

It is a tool for tracking student performance and identifying areas where students need improvement.
The software consists of three main modules: the student information form, the verbal questions, and the quantitative questions.
The student information form collects data about the student's background, including their name, age, address, and contact information.
The verbal questions module assesses the student's understanding of the material. It consists of a series of questions that the student must answer. The questions are designed to test the student's knowledge of the material.
The quantitative questions module assesses the student's understanding of the material. It consists of a series of questions that the student must answer. The questions test the student's ability to apply the material.

1.3 Software context

The software in this project is intended to help businesses or individuals analyze customer feedback to improve customer satisfaction. The software provides a chatbot interface or a graphical user interface (GUI) for users to input customer feedback. The software then analyzes the feedback and provides suggestions for improvement.
There are several strategic issues to consider when using this software. First, it is important to ensure that customer feedback is collected promptly. This feedback should be collected regularly, such as daily or weekly, to ensure that the software can provide accurate and up-to-date suggestions for improvement.
Second, it is important to ensure that customer feedback is collected from a representative sample of customers. This representative sample should be representative of the population of customers that the business or individual serves.
Finally, the software must be used with other customer satisfaction initiatives. For example, if the software suggests that a certain type of customer is unhappy, it is important to take action to address this issue. The software should not be used as a replacement for other customer satisfaction initiatives but rather as a supplement.

1.4 Major constraints

The first constraint is that the software must be able to analyze the text input accurately by the user to determine whether or not it contains swear words. This will require careful design and implementation to avoid false positives and negatives. The software will need to be designed to recognize common swear words in various languages and common ways to disguise swear words. In addition, the software will need to be able to handle different dialects and regional variations of swear words. False positives could result in the software incorrectly identifying a harmless word or phrase as a swear word, while false negatives could result in the software missing a swear word in the text.
The second constraint is that the software must be able to provide an appropriate response to the user based on the analysis of the input text. This will require the development of a chatbot or other interface that can communicate with the user naturally. The software will need to generate a response appropriate for the context in which the swear word was used. For example, if the swear word is used in a positive or neutral context, the software should provide a positive or neutral response. However, if the swear word is used in a negative or offensive context, the software should provide a negative or offensive response.
The third constraint is that the software must be able to run on various platforms, including Windows, macOS, and Linux. This will require careful testing on each platform to ensure

compatibility. The software will need to be designed to work with the different file systems and libraries used on each platform. In addition, the software will need to be tested on various hardware configurations to ensure that it can run on a wide range of devices.

2.0 Data Design

There are three data structures used in this project:

- The Student Information Form.
- The Verbal data.
- The Quantitative.

2.1 Data structures

- The Student Information Form data structure stores the details of the students.
- The Verbal data structure stores the questions and answers for the verbal section.
- The Quantitative data structure stores the questions and answers for the quantitative section.

2.2 Database Description

The first database is the Student Information Form. This database includes the details of the students.

The second database is the Verbal. This database includes the questions and answers for the verbal section.

The third database is Quantitative. This database includes the questions and answers for the quantitative section.

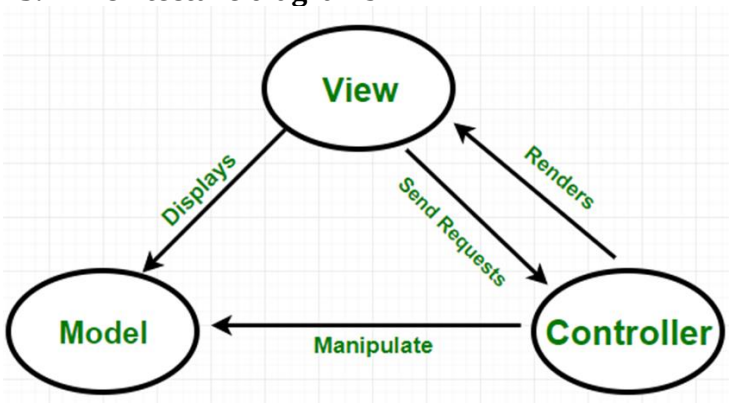
3.0 Architectural and component-level design

The software architecture of the above project is based on the Model-View-Controller (MVC) pattern. The Model represents the data, the View represents the graphical user interface, and the Controller represents the logic that handles user input and updates the Model.

The Model is responsible for storing and providing methods to access and update the data. The View is responsible for displaying the data and providing ways for the user to interact with the data. The Controller is responsible for handling user input and updating the Model and View accordingly.

The Model-View-Controller pattern is a good choice for this project because it separates the concerns of the different parts of the software and makes the code more modular and easier to understand.

3.1 Architecture diagrams



The logical view would concern the system's behavior and how it interacts with users. The physical theory would be concerned with the hardware and software components that make up the system. The development view would be concerned with the process of developing the plan.

From a logical perspective, the system is designed to analyze user input and generate responses accordingly. The system interacts with users through a chat interface. The system can understand user input and create replies, therefore.

From a physical perspective, the system is implemented using a combination of hardware and software components. The hardware components include a computer with a chat interface. The software components include a chatbot application and a natural language processing module.

From a development perspective, the system is developed using a spiral model of software development. The first step is to create a prototype of the system. This prototype is then evaluated and refined. The final design is then developed and tested.

3.2 Description of Components

The Model is responsible for storing and retrieving data. In the project context, the Model would be responsible for storing and retrieving information about students, such as their names, contact information, and grades.

The View is responsible for displaying data to the user. In the project context, the View would be responsible for displaying information about students to the user, such as their name, contact information, and grades.

The Controller is responsible for handling user input and updating the Model and View accordingly. In the context of the above project, the Controller would be responsible for managing user input and updating the Model and View accordingly. For example, if the user inputs a new student's name, the Controller will update the Model and View to reflect this change.

3.2.1 Model

3.2.1.1 Interface description

Responsible for storing and retrieving information about students, such as their names, contact information, and grades.

3.2.2 View

3.2.2.1 Interface description

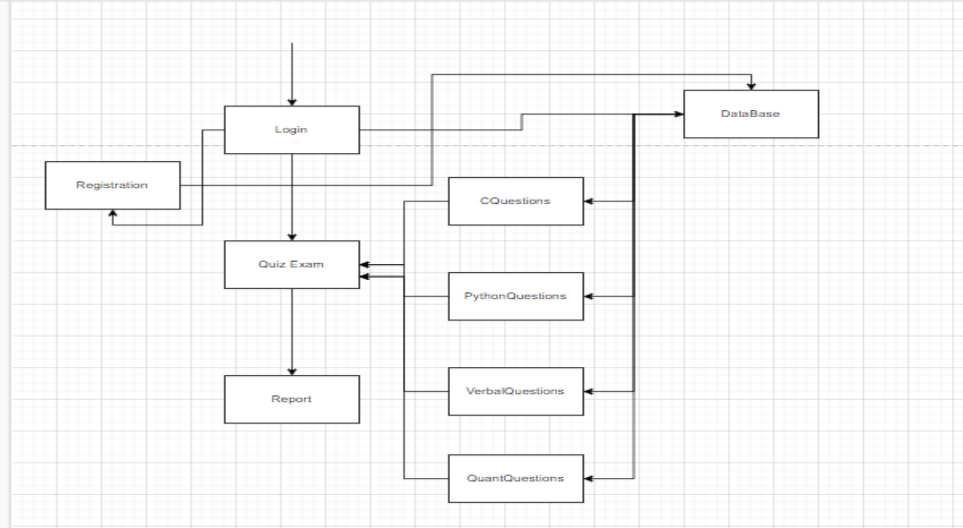
The view would be responsible for displaying information about students to the user, such as their name, contact information, and grades.

3.2.3 Controller

3.2.3.1 Interface description

Responsible for handling user input and updating the Model and View accordingly. In the context of the above project, the Controller would be responsible for managing user input and updating the Model and View accordingly. For example, if the user inputs a new student's name, the Controller will update the Model and View to reflect this change.

3.2.3.2 Static models



3.2.3.3 Dynamic models

Activity diagrams, sequential diagrams, state diagrams,

3.3 External Interface Description

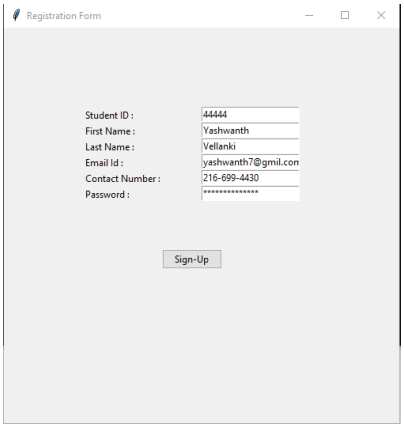
The software's external interface is the graphical user interface (GUI). The user can input data into the system through the GUI. The system then processes the data and displays the results on the GUI. The user can also view the results of the processing on the GUI.

The system also has an interface to the chatbot. The user can input data into the system through the chatbot. The system then processes the data and displays the results on the chatbot. The user can also view the results of the processing on the chatbot.

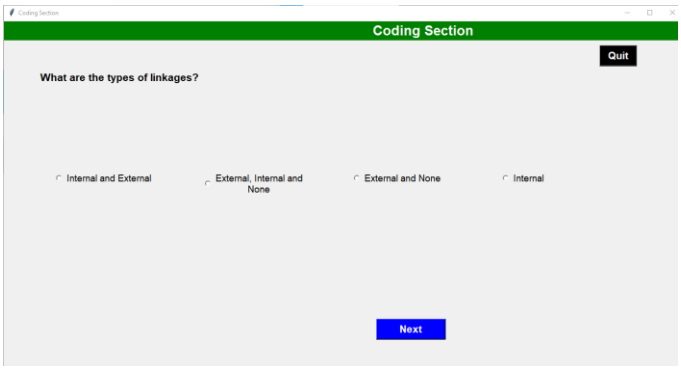
4.0 User interface design

4.1 Description of the user interface

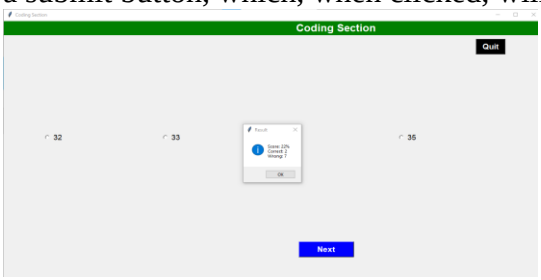
The student information form allows the user to input details about the student, such as name, age, gender, etc. The document also contains a submit button, which will save the student's information to a database when clicked.



The verbal section contains a series of questions that the student must answer. The questions are designed to test the student's understanding of the English language. The team also includes a submit button, which, when clicked, will save the student's answers to the database.



The quantitative section contains a series of questions that the student must answer. The questions are designed to test the student's understanding of mathematical concepts. The team also includes a submit button, which, when clicked, will save the student's answers to the database.



4.2 Interface design rules

• The user interface should be designed using a standard GUI toolkit such as Tkinter.
• The user interface should be designed using standard widgets such as buttons, labels, text fields, etc.
• The user interface layout should be simple and easy to understand.
• The user interface should be designed to be easy to use and navigate.

5.0 Restrictions, limitations, and constraints

One of the most important design issues is the need to create an easy and effective user interface. Another design issue is the need to ensure that the software can handle large data. Finally, the design issue of security is also important, as the software must be able to protect the data it stores from unauthorized access.

Group 9

Nithin Aryan Agrishetty	**2837033**
Jessica mathi	**2843649**
Sahithi Varma Raghavaraju	**2837094**
Halaa Pranavi Vangara	**2837402**