**CHAPTER-1**

**INTRODUCTION**

The main aim of our application (Visual Amigo) is to act as an assistant to visually challenged or impaired people. It includes three major modules namely Object Identification from Images using Camera, Live and continuous sharing of location to their friends or relatives and personal voice assistant.

To easily identify the objects around them a user can open the application and capture the image and our application will tell the object (most likely) with sound so that a visually impaired person can easily understand.

After seeking the required permission which ensures security, user location will be continuously updated in the database (till there is data connectivity) so that a relative or friend of visually impaired person can easily track them.

At last an assistant is also included which can call, open the desired chats and applications, change some settings like Wi-Fi and Ringer modes, find nearest places (restaurants, hospitals, etc..), make YouTube search, download apps and Google anything you want with voice commands.

**CHAPTER-2**

**Existing system and Proposed system**

To Edit

**CHAPTER-3**

**Feasibility study**

A feasibility study involves taking a judgment call on whether a project is doable. The two criteria to judge feasibility are cost required andvalue to be delivered*.* A well-designed study should offer a historical background of the business or project, a description of the product or service, accounting statements, details of operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, such studies precede technical development and project implementation.

A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions.

**3.1 Technical Feasibility**

Technical feasibility involves evaluation of the hardware and the software requirements of the proposed system.

In this project, the technologies involved are Google Cloud Vision API, Google Speech to Text API and Android Studio are being used to develop the Mobile Application. These play a major role in fulfilling the project requirements with good accuracy and within minimal time.

**3.2 Economic Feasibility**

Economic Feasibility helps in assessing the viability, cost, and benefits associated with projects before financial resources are allocated. This assessment typically involves a cost/ benefits analysis of the project.

The application is so designed that it requires minimal cost and eliminates costs as there would minimal need for manual work. The technologies used helps in understanding the user without any investment.

**3.3 Legal Feasibility**

The proposed system doesn’t conflict with legal requirements like data protection acts or social media laws. It ensures legal data access and gives prominence to data security.

**3.4 Operational Feasibility**

The application involves design-dependent parameters such as reliability, maintainability, supportability, usability, disposability, sustainability, affordability, and others.

//To Edit

It minimises the drawbacks of the current system by building an application that automatically resolves the user queries and helps to analyse the user data.

**3.5 Scheduling Feasibility**

The project development took place in timely process by understanding time schedules of the project and maintains good time line for project development.

**CHAPTER-4**

**System Analysis**

**CHAPTER-5**

**SYSTEM DESIGN**

**5.1 UML DIAGRAMS**

The Unified Modelling Language allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

• **User Model View**

- This view represents the system from the user’s perspective.

- The analysis representation describes a usage scenario from the end-user’s perspective.

• **Structural Model View**

- In this model the data and functionality are arrived from inside the system.

- This model view models the static structures.

• **Behavioural Model View**

It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structure elements described in the user model and structural model view.

• **Implementation Model View**

In this the structural and behavioural as parts of the system are represented as they are to be built.

UML is specifically constructed through two different domains they are:

* UML Analysis modelling, this focuses on the user model and structural model views of the system.
* UML design modelling, which focuses on the behavioral modelling, implementation modelling and environmental modelling.
* Use case Diagrams represents the functionality of the system from user’s point of view.
* Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with system. Examples of actors include users like administrator, bank customer…. etc., or another system like central database.

**5.2 CLASS DIAGRAM**

A class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object-oriented modelling. It is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

• The top part contains the name of the class. It is printed in Bold, centered and the first letter capitalized.

• The middle part contains the attributes of the class. They are left aligned and the first letter is lower case.

• The bottom part gives the methods or operations the class can take or undertake. They are also left aligned and the first letter is lower case.

In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modelling, the classes of the conceptual design are often split into a number of subclasses.

A screenshot of a map

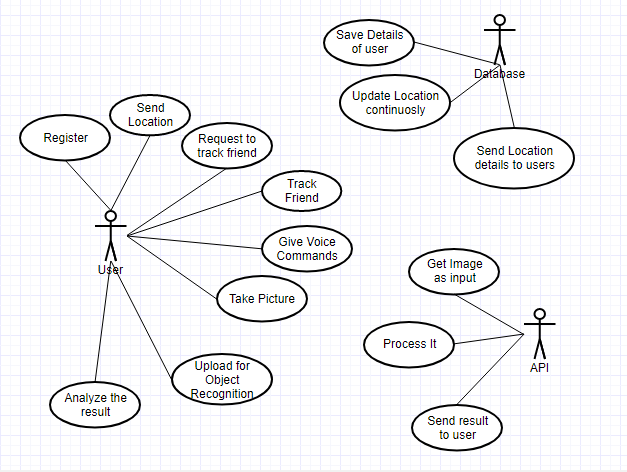
Description generated with very high confidence

**FIG 5.1 CLASS DIAGRAM**

// Add Description

**5.3 USECASE DIAGRAM**

A Use case diagram at its simplest is a representation of a user’s interaction with the system and depicting the specifications of use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



**FIG 5.2 USE CASE DIAGRAM**

// Add Description

**5.4 SEQUENCE DIAGRAM**

A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. Sequence diagrams consists of two parts flows of control and scanlines: Where flow of control is used in the diagram to check the sequence and scanlines is the continuous part until the particular sequence in the sequence diagram.

A close up of a map

Description generated with very high confidence

**FIG 5.3 SEQUENCE DIAGRAM**

// Add Description

**5.5 ACTIVITY DIAGRAM**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control. Activity diagrams are constructed from a limited number of shapes, connected with arrows.

The most important shape types:

• rounded rectangles represent actions;

• diamonds represent decisions;

• bars represent the start (split) or end (join) of concurrent activities;

• A black circle represents the start (initial state) of the workflow;

• An encircled black circle represents the end (final state).

Arrows run from the start towards the end and represent the order in which activities happen. Activity diagrams may be regarded as a form of flowchart. Typical flowchart techniques lack constructs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops. While in UML 1.x, activity diagrams were a specialized form of state diagrams in UML 2.x, the activity diagrams were deformalized to be based on Petri net-like semantics, increasing the scope of situations that can be modelled using activity diagrams. These changes cause many UML 1.x activity diagrams to be interpreted differently in UML 2.x.

A close up of a map

Description generated with very high confidence

**FIG 5.4 ACTIVITY DIAGRAM**

// Add Descriptions