

# DATING APPLICATION

*Submitted by*

**TINA KASHYAP RA2211028010226**  
**PARINITA GARG RA2211028010228**  
**LUVNEET VERMA RA2211028010221**  
**MINHAJ KHAN RA2211028010223**

*Under the Guidance of*

**Dr .S. MURUGAANANDAM**  
Associate Professor , CCAI  
Department of NETWORKING and COMMUNICATION

*In partial satisfaction of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**  
**in**  
**COMPUTER SCIENCE ENGINEERING**  
**with specialization in CLOUD COMPUTING**



**NETWORKING AND COMMUNICATION**  
**COLLEGE OF ENGINEERING AND TECHNOLOGY**  
**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  
**KATTANKULATHUR - 603203**

**MAY 2023**

**SRM INSTITUTION OF SCIENCE AND TECHNOLOGY  
KATTANKULATHUR-603203**

**BONAFIDE CERTIFICATE**

Certified that this Project Report titled “**DATING APPILICATION**” is the bonafide work done by **TINA KASHYAP RA2211028010226, PARINITA GARG RA2211028010228, LUVNEET VERMA RA2211028010221, MINHAI KHAN RA2211028010223** who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

**SIGNATURE**

**DR S.MURUGAANANDAM**

**OODP – Course Faculty**

Associate Professor, CCAI

Department of NETWORKING and Communication

**SRMIST**

**SIGNATURE**

**Dr ANNAPURANI**

**Head of the Department**

Department of NWC

**SRMIST**

## TABLE OF CONTENTS

S.No	CONTENTS	PAGE NO
1.	Problem Statement	
2.	Modules of Project	
3.	Diagrams	
	a. Use case Diagram	
	b. Class Diagram	
	c. Sequence Diagram	
	d. Collaboration Diagram	
	e. State Chart Diagram	
	f. Activity Diagram	
	g. Package Diagram	
	h. Component Diagram	
	i. Deployment Diagram	
4.	Code/Output Screenshots	
5.	Conclusion and Results	
6.	References	

## ABSTRACT

In recent years, dating applications have become increasingly popular as a means for individuals to connect and find potential romantic partners. This abstract presents the design and development of a dating application using Object-Oriented Analysis and Design (OOAD) principles.

The primary objective of this project is to create a user-friendly and efficient dating application that leverages the power of OOAD to provide a seamless and engaging user experience. The application incorporates various OOAD concepts such as inheritance, encapsulation, and polymorphism to ensure a modular and maintainable codebase.

The design process begins with requirements gathering, where user needs and expectations are identified. This information is used to develop a conceptual model, followed by the creation of use cases to define the application's functionalities. The use cases are then translated into class diagrams, illustrating the relationships and interactions between different entities in the system. The dating application utilizes an object-oriented programming language, such as Java or Python, to implement the design. The application architecture is structured using object-oriented design patterns, promoting code reusability and extensibility. The Model-View-Controller (MVC) pattern is commonly employed to separate the application's logic, user interface, and data management components.

Key features of the dating application include user profile creation, browsing and matching algorithms, real-time messaging, and privacy settings. The application employs OOAD principles to ensure the separation of concerns and modularity of these features, allowing for easy maintenance and future enhancements.

Throughout the development process, rigorous testing methodologies, such as unit testing and integration testing, are employed to verify the correctness and robustness of the application. Additionally, user feedback and usability testing are crucial in refining the application's design and user experience.

The final dating application showcases the benefits of using OOAD in its design and development. It offers users a reliable, feature-rich, and intuitive platform to connect with potential partners, while providing developers with a well-structured and maintainable codebase.

## INTRODUCTION

A dating application is a software application designed to help people find romantic partners. It typically involves creating a profile with personal information and preferences, searching for potential matches based on specific criteria, and communicating with other users through messaging or other means. Here are some features that could be included in a dating application: User profiles: Users would create a profile with personal information such as name, age, location, and interests. They would also be able to upload photos and write a bio.

1. **Matching algorithm:** The application would use a matching algorithm to suggest potential matches based on the user's preferences and the preferences of other users.
2. **Search filters:** Users would be able to search for potential matches based on various criteria, such as age, location, interests, and relationship status.
3. **Messaging:** Users would be able to communicate with each other through messaging, either within the application or through another messaging platform.
4. **Video or voice calling:** The application could include a feature for users to make video or voice calls with each other.
5. **Social media integration:** The application could integrate with social media platforms to allow users to import their social media profiles and connect with other users who have similar interests.
6. **Privacy and safety features:** The application could include features to ensure user privacy and safety, such as the ability to block or report other users, and options for keeping certain information private.
7. **Premium features:** The application could offer premium features for a fee, such as the ability to see who has viewed your profile or the ability to send virtual gifts to other users.

These features could be implemented using various programming languages and technologies, such as Java, Python, PHP, or Ruby on Rails, and could be hosted on various platforms, such as the web, mobile devices, or desktop computers.

## ABOUT OOPD

Object-oriented design started right from the moment computers were invented. Programming was there, and programming approaches came into the picture. Programming is basically giving certain instructions to the computer.

At the beginning of the computing era, programming was usually limited to machine language programming. Machine language means those sets of instructions that are specific to a particular machine or processor, which are in the form of 0's and 1's. These are sequences of bits (0100110...). But it's quite difficult to write a program or develop software in machine language. It's actually impossible to develop software used in today's scenarios with sequences of bits. This was the main reason programmers moved on to the next generation of programming languages, developing assembly languages, which were near enough to the English language to easily understand. These assembly languages were used in microprocessors. With the invention of the microprocessor, assembly languages flourished and ruled over the industry, but it was not enough. Again, programmers came up with something new, i.e., structured and procedural programming.

The Object-Oriented Programming (OOP) Approach –

The OOP concept was basically designed to overcome the drawback of the above programming methodologies, which were not so close to real-world applications. The demand was increased, but still, conventional methods were used. This new approach brought a revolution in the programming methodology field.

Object-oriented programming (OOP) is nothing but that which allows the writing of programs with the help of certain classes and real-time objects. We can say that this approach is very close to the real-world and its applications because the state and behaviour of these classes and objects are almost the same as real-world objects.

Let's go deeper into the general concepts of OOP, which are given below:

What Are Class & Object?

It is the basic concept of OOP; an extended concept of the structure used in C. It is an abstract and user-defined data type. It consists of several variables and functions. The primary purpose of the class is to store data and information. The members of a class define the behaviour of the class. A class is the blueprint of the object, but also, we can say the implementation of the class is the object. The class is not visible to the world, but the object is.

#### Data Abstraction –

Abstraction refers to the act of representing important and special features without including the background details or explanation about that feature. Data abstraction simplifies database design.

#### Physical Level:

It describes how the records are stored, which are often hidden from the user. It can be described with the phrase, “block of storage.”

#### Logical Level:

It describes data stored in the database and the relationships between the data. The programmers generally work at this level as they are aware of the functions needed to maintain the relationships between the data.

#### View Level:

Application programs hide details of data types and information for security purposes. This level is generally implemented with the help of GUI, and details that are meant for the user are shown.

#### Encapsulation –

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit, e.g., a class in Java. This concept is often used to hide the internal state representation of an object from the outside.

#### Inheritance –

Inheritance is the ability of one class to inherit capabilities or properties of another class, called the parent class. When we write a class, we inherit properties from other classes. So when we create a class, we do not need to write all the properties and functions again and again, as these can be inherited from another class that possesses it. Inheritance allows the user to reuse the code whenever possible and reduce its redundancy.

#### Polymorphism –

Polymorphism is the ability of data to be processed in more than one form. It allows the performance of the same task in various ways. It consists of method overloading and method overriding, i.e., writing the method once and performing a number of tasks using the same method name.

#### Advantages of OODP –

It models the real world very well.

With OOP, programs are easy to understand and maintain.

OOP offers code reusability. Already created classes can be reused without having to write them again.

OOP facilitates the quick development of programs where parallel development of classes is possible.

With OOP, programs are easier to test, manage and debug.

#### Disadvantages of OOP –

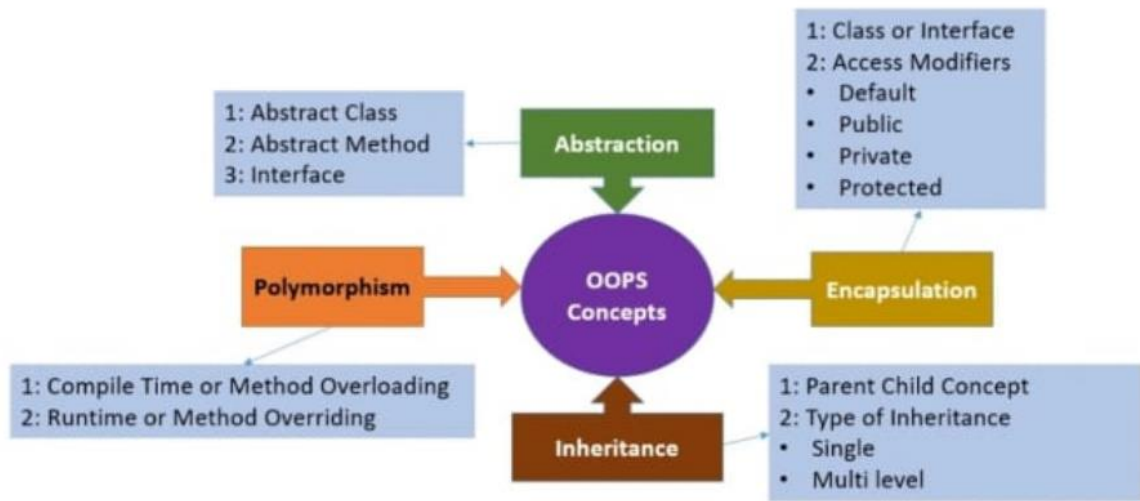
With OOP, classes sometimes tend to be over-generalized.

The relations among classes become superficial at times.

The OOP design is tricky and requires appropriate knowledge. Also, one needs to do proper planning and design for OOP programming.

To program with OOP, the programmer needs proper skills such as design, programming, and thinking in terms of objects and classes, etc.





**OOPS Concepts**

## MODULES OF THE PROJECT

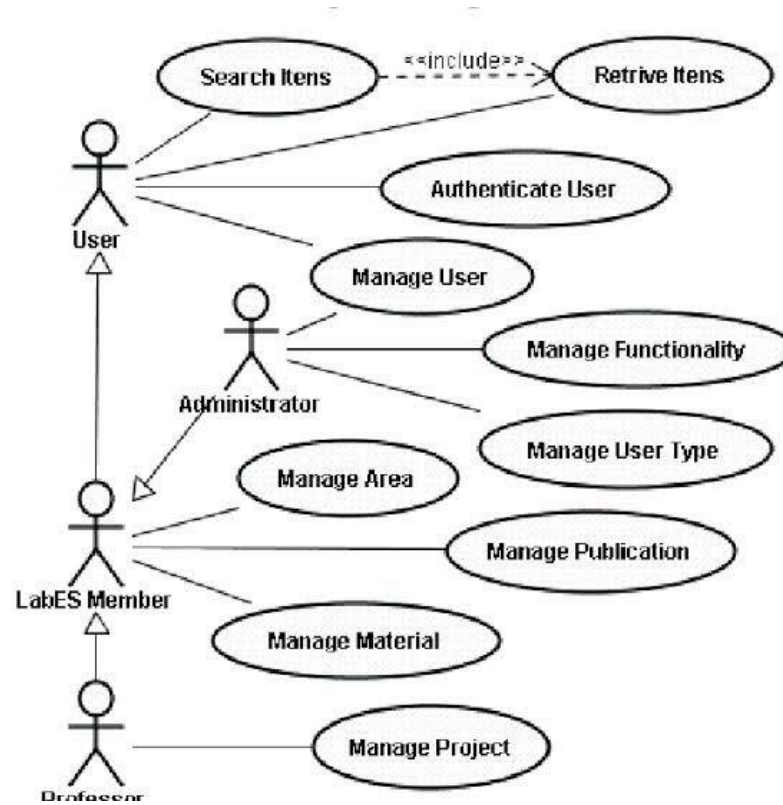
A dating application can be broken down into various modules or components that work together to provide a complete and functional application. Here are some of the main modules that could be included in a dating application:

1. User authentication and profile management module: This module would handle user authentication and account creation, as well as managing user profiles and preferences. It would allow users to create and edit their profiles, and also manage their preferences for potential matches.
2. Matching algorithm module: This module would be responsible for matching users based on various criteria, such as age, location, interests, and other factors. It would use machine learning algorithms and data analytics to suggest potential matches to users.
3. Search and filter module: This module would allow users to search for potential matches based on various criteria, such as age, location, interests, and other factors. It would also allow users to filter their search results to find matches that meet their specific criteria.
4. Messaging module: This module would enable users to communicate with each other through messaging, either within the application or through another messaging platform. It would include features such as real-time chat, message history, and the ability to block or report other users.
5. Media management module: This module would handle the management of media files such as photos and videos uploaded by users. It would include features such as image and video compression, file storage, and retrieval.
6. Payment and billing module: This module would handle payment processing and billing for premium features, such as the ability to send virtual gifts or see who has viewed your profile.
7. Reporting and analytics module: This module would provide analytics and reporting features to track the usage of the application, user engagement, and other metrics. It would help identify areas for improvement and make data-driven decisions for future development.

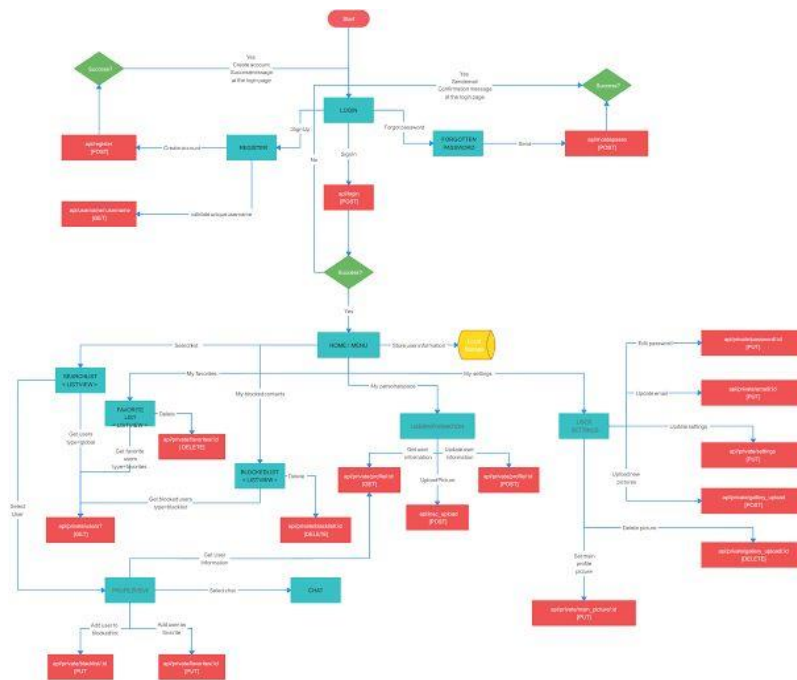
These modules could be implemented using various programming languages and technologies, such as Java, Python, PHP, or Ruby on Rails, and could be hosted on various platforms, such as the web, mobile devices, or desktop computers.

# DIAGRAMS

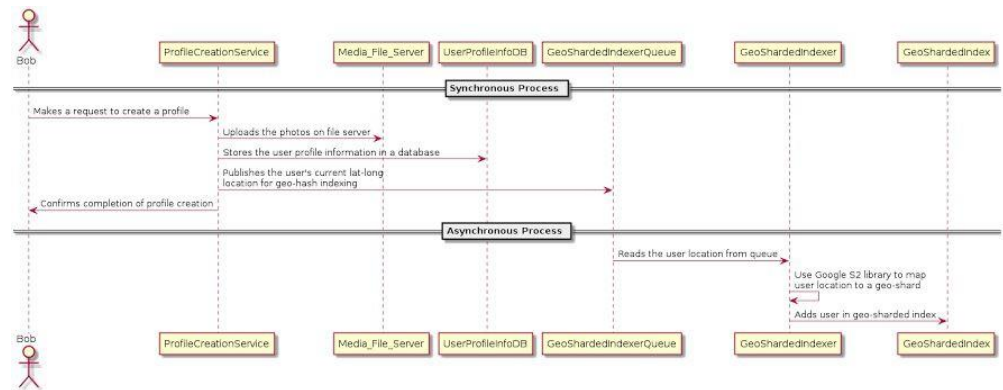
## 1. USE CASE



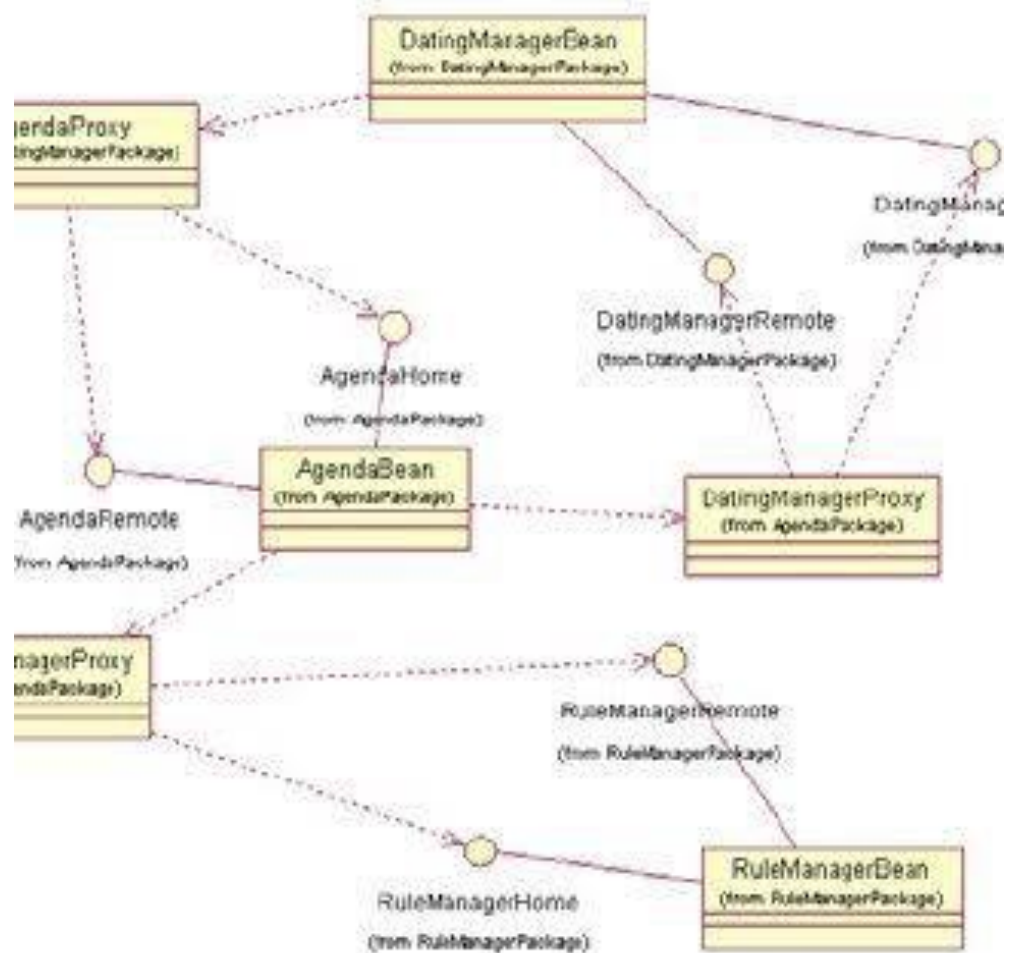
## 2. CLASS DIAGRAM



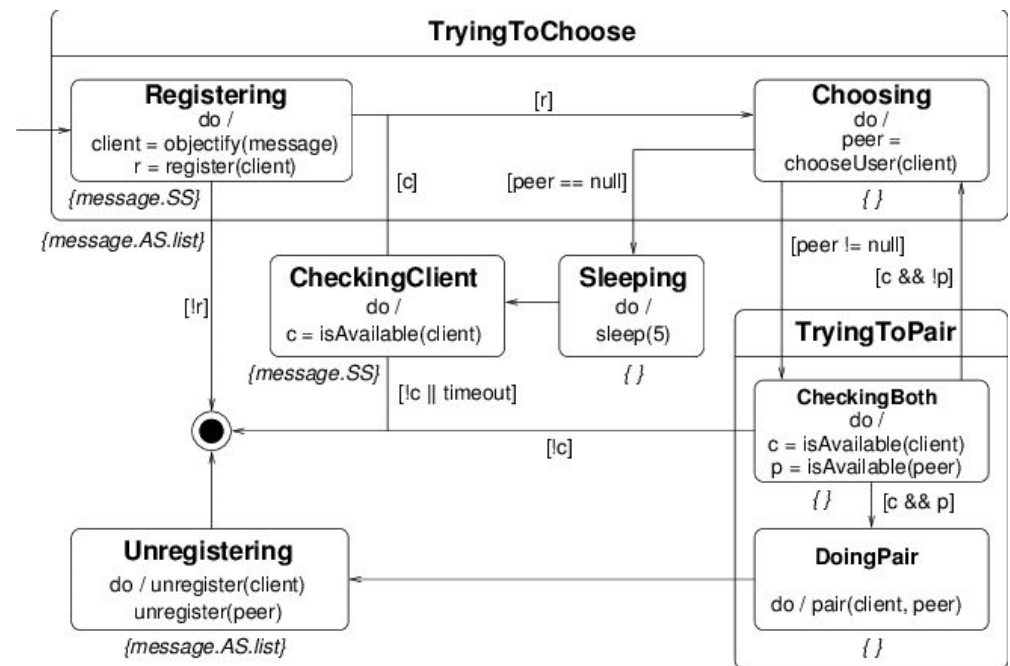
### 3. SEQUENCE DIAGRAM



## 4. COLLABORATION DIAGRAM

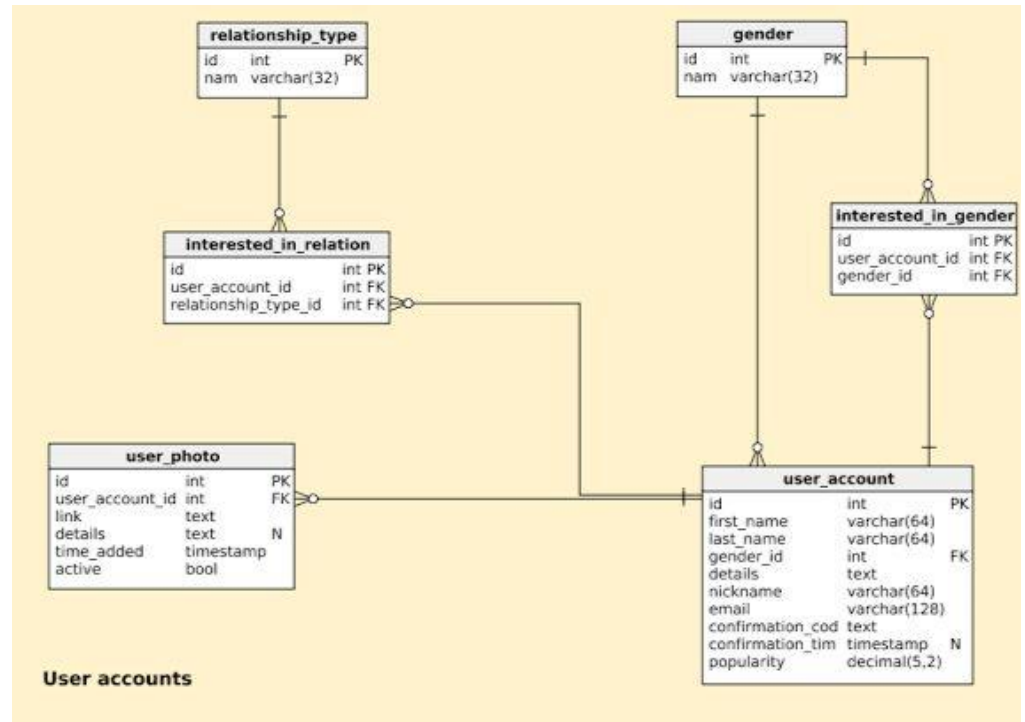


## 5. CHART DIAGRAM

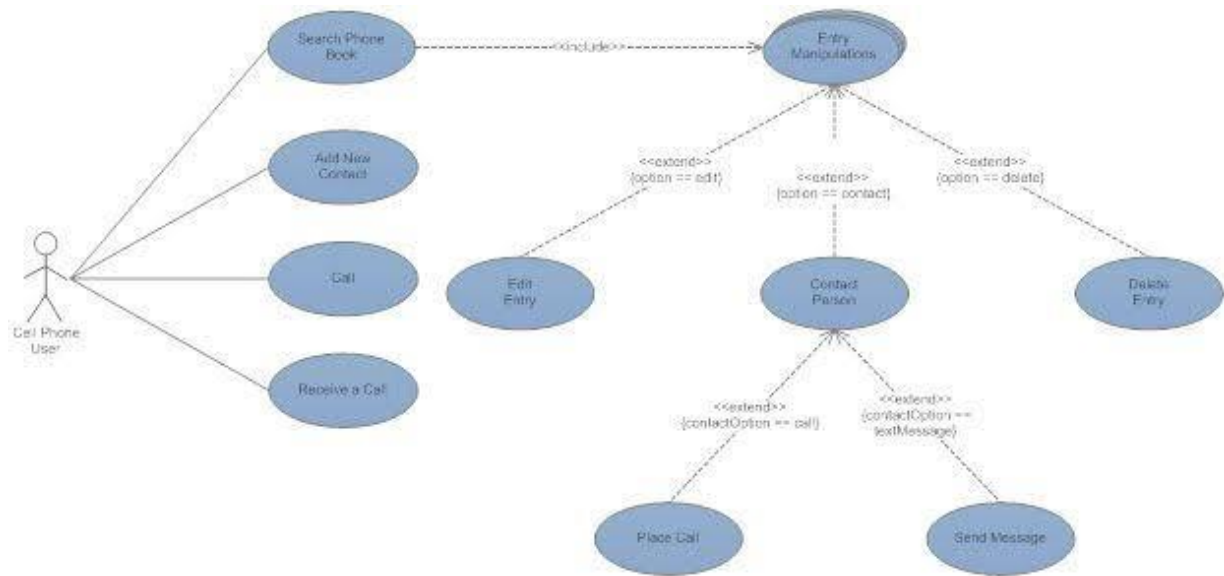




## 6. PACKAGE DIAGRAM



## 7. COMPONENT DIAGRAM



## CODE

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

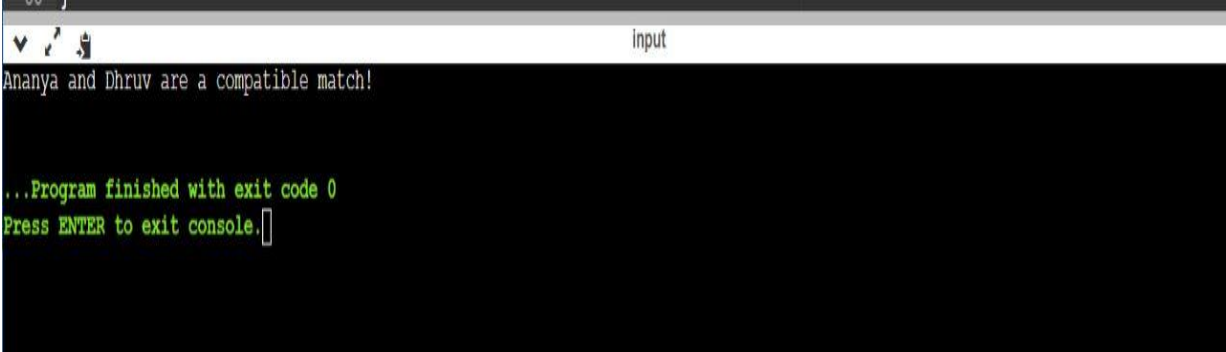
// User class
class User {
public:
    string name;
    int age;
    string gender;
    string interest;
    string bio;

    User(string name, int age, string gender, string interest, string bio) {
        this->name = name;
        this->age = age;
        this->gender = gender;
        this->interest = interest;
        this->bio = bio;
    }
};

// Match function that returns true if two users are compatible
bool match(User user1, User user2) {
    if (user1.gender == user2.interest && user1.interest == user2.gender && user1.age >= 18
    && user2.age >= 18) {
        return true;
    }
    return false;
}
```

```
int main() {  
    // Sample users  
    User user1("Alice", 25, "female", "male", "I love hiking and trying new foods.");  
    User user2("Bob", 30, "male", "female", "I'm a huge fan of sci-fi movies and video  
games.");  
    User user3("Charlie", 22, "male", "male", "I'm into fitness and love traveling.");  
  
    // Vector of users  
    vector<User> users = {user1, user2, user3};  
  
    // Match users  
    for (int i = 0; i < users.size(); i++) {  
        for (int j = i+1; j < users.size(); j++) {  
            if (match(users[i], users[j])) {  
                cout << users[i].name << " and " << users[j].name << " are a compatible match!\n";  
            }  
        }  
    }  
  
    return 0;  
}
```

## EXPERIMENT RESULTS AND OUTPUTS



A screenshot of a terminal window with a dark background. The window title bar at the top is light gray and contains the text "Input" on the right side. The terminal displays the following text in white: "Ananya and Dhruv are a compatible match!". Below this, in green text, it says "...Program finished with exit code 0" and "Press ENTER to exit console." followed by a small white cursor icon.

```
Input
Ananya and Dhruv are a compatible match!

...Program finished with exit code 0
Press ENTER to exit console.
```

## REFERENCES

1. GOOGLE
2. <https://www.geeksforgeeks.org/oops-object-oriented-design/>
3. Github
4. W3schools
5. codeforhours