**A Project Report**

**of**

**AI-Driven Privacy Scoring And Sanitization Of User Uploaded Photo**

A Project Report

Submitted in partial fulfilment of the

Requirements for the Award of the Degree of

**BACHELOR OF SCIENCE (COMPUTER SCIENCE)**

**By**

**Ms. Hiteshree Ganesh Patel (28)**

**Seat Number:**

**Under the esteemed guidance of**

**Miss. Jigna Bhansali**

**Assistant Professor**



**St. John College of Humanities and Sciences**

**(Autonomous)**

**(Affiliated to Mumbai University)**

**PALGHAR-MAHARASHTRA-401404**

**(SEM V) 2025-2026**

**ST. JOHN COLLEGE OF HUMANITIES & SCIENCES**

**(Autonomous)**

**(Affiliated to University of Mumbai)**

**PALGHAR-MAHARASHTRA-401404**

**DEPARTMENT OF COMPUTER SCIENCE**



**CERTIFICATE**

This is to certify that the project entitled, AI-Driven Privacy Risk Scoring and Sanitization of

User Uploaded photos, is bonafide work of Ms. HITESHREE PATEL bearing

Seat. No: ( ) submitted in partial fulfilment of the requirements for the award of degree BACHELOR OF SCIENCE in COMPUTER SCIENCE from University of Mumbai.

**Project Guide External Examiner Internal Examiner**

**Date: HOD College Seal**

**ABSTRACT**

With the rapid growth of social media and digital communication, users frequently share personal

images online without fully realizing the associated privacy risks. These images may contain sensitive

content such as identifiable faces, ID cards, license plates, or inappropriate material that can compromise

individual security. This project presents a Photo Privacy Scoring and Protection Web Application that

automatically evaluates and enhances the privacy of user-uploaded images.

The system leverages YOLOv8 pre-trained deep learning models for detecting sensitive regions,

including human faces, textual identifiers, and NSFW elements. A privacy risk scoring mechanism is then

applied, assigning a score based on the severity and type of detected content. To mitigate risks, the system

provides automatic sanitization options such as blurring, masking, or removing sensitive areas.

Developed using a Flask-based backend with integrated computer vision and image-processing

techniques (OpenCV, Pillow), the application offers a simple web interface where users can upload photos,

receive an instant privacy assessment, and download a sanitized version of their image. The prototype

demonstrates how deep learning can be applied to improve data protection, content moderation, and

online privacy awareness, making it a valuable tool for individuals as well as platforms that handle user-

generated content.

**ACKNOWLEDGMENT**

I would like to express my greatest gratitude to the people who have helped and supported

me throughout my project. I am grateful to all my teachers for their continuous support for the

project from initial advice and contacts in the early stages of conceptual inception and through

ongoing advice and encouragement to this day. I would like to express my special thanks to our

principal(name) who gave me the golden opportunity to do this wonderful project on the topic ‘AI-

Driven Privacy Risk Scoring and Sanitization of User Uploaded photos’, which also helped me in

doing a lot of research and I came to know about so many new things. I am really thankful to

them. I would also like to thank my project guide Miss. Jigna Bhansali

**MS. Hiteshree Patel**

**DECLARATION**

I hereby declare that the project entitled, “AI-Driven Privacy Risk Scoring and Sanitization of

User Uploaded photos ”, has not been in any case duplicated to submit to any other university for

the award of any degree. To the best of our knowledge other than us, no one has submitted to any

other university. The project is done in partial fulfilment of the requirements for the award of degree

of BACHELOR OF SCIENCE (COMPUTER SCIENCE) to be submitted as a fifth semester

project as part of our curriculum.

**MS. Hiteshree Patel**

**TABLE OF CONTENTS**

**INTRODUCTION .......................................................................................................................................1**

**1.1 Background ......................................................................................................................................2**

**1.2 Objective:..........................................................................................................................................4**

**1.3 Purpose, Scope, and Applicability..................................................................................................6**

1.3.1 Purpose .....................................................................................................................................6

1.3.2 Scope ........................................................................................................................................6

1.3.3 Applicability .............................................................................................................................6

**1.4 Achievements....................................................................................................................................7**

**1.5 Organisation of Report....................................................................................................................8**

**SYSTEM ANALYSIS ..................................................................................................................................9**

**2.1 Existing system.................................................................................................................................9**

2.1.1 Limitations of existing systems ................................................................................................9

**2.2 Proposed systems ...........................................................................................................................10** **2.3 RequirementAnalysis...................................................................................................................10** **2.4 HardwareRequirements................................................................................................................12**

**2.5 Software Requirements .................................................................................................................12**

**2.6 Justification of selection of Technology .......................................................................................13**

**REQUIREMENT ANALYSIS .................................................................................................................14**

**3.1 Problem definition .........................................................................................................................14**

**3.2 Requirements Specification ..........................................................................................................16**

**3.3 Planning and Scheduling...............................................................................................................17**

3.3.1 Project Planning......................................................................................................................17

3.3.2 Project Scheduling..................................................................................................................18

**3.4 Software and Hardware Requirements .......................................................................................19** **3.5 Preliminary Product Description................................................................................................. 20**

**3.6 Conceptual Models ........................................................................................................................22**

3.6.1 ER Diagram ............................................................................................................................ 22 3.6.2 Activity Diagram .................................................................................................................... 23 3.6.3 Class Diagram ......................................................................................................................... 24

3.6.4 Use Case Diagram ..................................................................................................................25

**SYSTEM DESIGN.....................................................................................................................................26**

**4.1 Basic Modules.................................................................................................................................26**

**4.2 Data Design.....................................................................................................................................28**

4.2.1 Schema Design .......................................................................................................................28

4.2.2 Data Integrity and Constraints................................................................................................33

**4.3 Procedural Design..........................................................................................................................35**

4.3.1 Logic Diagrams ......................................................................................................................35

4.3.2 Data Structures........................................................................................................................36

4.3.3 Algorithm Design ...................................................................................................................37

**4.4 User Interface Design ....................................................................................................................39**

**4.5 Security Issues ................................................................................................................................41**

**4.6 Test Cases .......................................................................................................................................42**

**IMPLEMENTATION AND TESTING ...................................................................................................... 43**

**5.1 Implementation Approaches .................................................................................................... 43**

**5.2 Code .......................................................................................................................................... 43**

**5.3 Coding Details and Code Efficiency ........................................................................................ 82**

5.3.1 Code Efficiency .................................................................................................................. 82

**5.4 Testing Approach ..................................................................................................................... 82**

5.4.1 Unit Testing ........................................................................................................................ 82

5.4.2 Integrated Testing .............................................................................................................. 82

5.4.3 Beta Testing ....................................................................................................................... 82

**5.5 Modifications and Improvements ............................................................................................ 82**

**5.6 Test Cases ................................................................................................................................. 83**

**RESULTS AND DISCUSSION ............................................................................................................... 84**

**6.1 Test Reports .............................................................................................................................. 84**

**6.2 User Documentation ................................................................................................................. 84**

**CONCLUSIONS ..................................................................................................................................... 87**

**7.1 Conclusion ................................................................................................................................ 87**

7.1.1 Significance of the System ................................................................................................. 87

**7.2 Limitations of the System ........................................................................................................ 87**

**7.3 Future Scope of the Project ..................................................................................................... 87**

**7.4 References ................................................................................................................................ 88**

**LIST OF TABLES**

**\**

Table 1: **Planning and Scheduling**………………………………………………………………………17

Table 2 : **User**……………………………………………………………………………………………..28

Table 3 **: Period\_Cycles** ………………………………………………………………………………….28

Table 4 : **Disease Assessment** ……………………………………………………………………………29

Table 5 : **Cart**……………………………………………………………………………………………...29

Table 6: **Orders**…………………………………………………………………………………………...30 Table 7: **Products** ………………………………………………………………………………………...30

Table 8: **Appointments** …………………………………………………………………………………..31 Table 9: **Doctors** ………………………………………………………………………………………….31

Table 10: **Meal\_Plan** ……………………………………………………………………………………..32

Table 11: **Test Cases**………………………………………………………………………………………42

**LIST OF FIGURES**

Figure 1 **: Gantt Chart** ………………………………………………………………………………......18

Figure 2 : **ER Diagram**…………………………………………………………………………………...22

Figure 3 : **Activity Diagram** ……………………………………………………………………………..23 Figure 4 : **Class Diagram** ………………………………………………………………………………..24

Figure 5 : **Use Case Diagram** ……………………………………………………………………………25 Figure 6 : **Logic Diagram** ………………………………………………………………………………..35

Figure 7 : **User Interface Design** ………………………………………………………………………...39

**CHAPTER 1**

# INTRODUCTION

In today’s world, sharing photos online has become a part of our everyday life. Whether it’s posting on

social media, sending pictures to friends, or uploading documents for official purposes, images travel faster than ever before. But while this makes communication easy, it also comes with hidden risks. Many of us don’t realize that a single photo can reveal more than what we intend to share. A casual picture may show someone’s home address in the background, a car number plate on the street, or even an ID card lying on a desk. Once shared online, these details can be misused in ways that are difficult to control.

Most people rely on basic editing tools if they want to hide something in their photos, but this requires time,effort, and, more importantly, awareness of what needs to be hidden. Unfortunately, not everyone notices these small but important details before uploading. This is where the problem begins — privacy is often compromised, not because people want to share sensitive information, but because they don’t know it’s there.

The aim of this project is to create a system that helps users avoid these mistakes. The proposed system,

called Privacy Scoring and Sanitization of User Uploaded Photos, works in two simple steps. First, it

checks the photo using artificial intelligence and computer vision to detect sensitive details like faces,

license plates, documents, or text. Based on what it finds, it gives the photo a privacy score, which tells

the user how risky it is to share the image as it is. Second, it provides options to sanitize the photo by

blurring, masking, or hiding the risky areas. With this, users don’t just get a warning — they also get an

immediate solution.

The best part of this system is its simplicity. A user only needs to upload the photo; the system takes care

of the rest. It analyzes, scores, and suggests changes within seconds, making it both practical and easy to

use. Beyond personal use, this system can also be integrated into platforms like social media apps,

workplaces, or government portals to make privacy protection an automatic part of the photo-sharing

process.

This project is more than just a technical solution — it’s about building trust and safety in the digital space. By combining awareness (through privacy scoring) and action (through sanitization), it empowers users to share with confidence. In a time where personal data is more vulnerable than ever, this project takes a step toward ensuring that what we share online remains under our control.

## 1.1 Background

In the present digital era, photographs have become one of the most common and powerful ways of

communication. With the increasing use of smartphones, high-speed internet, and social media platforms,

people are constantly capturing and sharing moments of their lives online. Whether it is a casual selfie, a

family gathering, a travel update, or an official document upload, images are shared every second across

different platforms. While this trend makes interaction and expression easier, it also brings along hidden

threats to privacy and security.

What many people do not realize is that a single image can reveal far more information than intended. For

instance, a photo taken outside one’s home may expose the address or surroundings, a picture of an ID card

might display confidential details, and even a simple background could give clues about a person’s

workplace or location. Once such information is uploaded to the internet, it becomes very difficult to control

who can access it or how it might be used. This creates opportunities for cybercriminals, identity thieves,

and malicious actors who actively search for sensitive data hidden within images.

Currently, most users rely on manual methods to edit or sanitize their photos before sharing. They might

crop out certain areas or use simple tools to blur specific details. However, these methods have two major

limitations. First, they are time-consuming and require awareness — the user must know exactly what to

hide. Second, even careful users may miss certain sensitive elements that are not immediately obvious. As a result, private information may still slip through and get exposed.

The lack of an automated and reliable solution to this problem highlights the need for innovation. This is

where the idea of Privacy Scoring and Sanitization of User Uploaded Photos becomes relevant. The

project aims to bridge the gap between privacy awareness and practical action. By using artificial

intelligence and computer vision, the system can automatically analyze a photo, identify sensitive elements, and assign a privacy score that reflects how safe or risky the photo is to share. Instead of leaving the user

confused, the system provides a simple number that indicates the risk level.

Along with scoring, the system also provides sanitization options. This means that once sensitive areas are

identified, the system can automatically blur, mask, or pixelate them, giving the user a ready-to-share safe

version of the image. This reduces the burden on the user and ensures consistency and accuracy in protecting privacy.

The background of this project is rooted in the growing importance of data privacy in everyday life. With

governments, organizations, and individuals becoming more aware of cyber risks, tools like this can play a

vital role in creating safer digital habits. Not only can it help individuals who casually share photos online,

but it can also be applied in workplaces, corporate platforms, and official portals where document uploads

are common.

In short, the background of this project lies in the realization that while sharing photos has become

effortless, protecting privacy has not. The Privacy Scoring and Sanitization system provides a solution

that balances convenience with security, ensuring that users can continue sharing freely without

compromising their personal safety.

**1.2 Objective:**

The main aim of this project is to help users protect their privacy when sharing photos online. In today’s

digital world, people often upload images without realizing how much hidden information they might be

giving away. A simple picture can expose details like faces, documents, or even license plates, which may

be misused if it falls into the wrong hands. This project has been designed with the goal of making photo-

sharing safer, smarter, and more privacy-aware.

**The objectives of the system are:**

* **To detect sensitive elements in images**

The first and most important objective is to automatically identify sensitive objects or regions within

a photo. This includes faces, car number plates, ID documents, or any visible text that could carry

private information. Instead of expecting users to manually search for these details, the system takes

on this responsibility and ensures that nothing important is overlooked.

* **To assign a privacy score**

Simply detecting sensitive areas is not always enough. Users need a clear idea of how risky their

image really is. For this reason, the system assigns a privacy score to each photo. This score works

like a warning signal, telling the user whether their image is safe to share as it is, or whether it needs

sanitization. The score makes privacy risks easy to understand even for non-technical users.

* **To provide sanitization options**

Detection and scoring are only useful if the user can take action. Therefore, the system provides easy

sanitization options like blurring, masking, or pixelating the detected sensitive areas. The process is

fully automated, meaning that users don’t need to spend time editing their images manually. Within

seconds, they can download a safer, sanitized version of their photo.

* **To build a user-friendly platform**

Another important objective is to make the system accessible to everyone, regardless of their

technical skills. The platform has been designed with simplicity in mind — users only need to upload

their image, and the system handles the rest. This reduces effort and makes the tool practical for

everyday use.

* **To make the system adaptable for wider use**

While the project can directly help individuals, it has also been developed with a broader vision. The

system can be integrated into organizations, corporate platforms, and even social media networks to

promote safe photo sharing on a larger scale. By making the system adaptable, it can serve as a

privacy safeguard for both personal and professional needs.

In short, the objectives of this project go beyond just detecting risks. The goal is to create awareness

through privacy scoring, provide immediate solutions through sanitization, and make the entire

process simple and accessible. This ensures that users can share photos with confidence, knowing that their privacy is protected.

## 1.3 Purpose, Scope, and Applicability

### 1.3.1 Purpose

The main purpose of this project is to protect users from unintentionally exposing personal details while

sharing photos online. Many people share pictures without realizing that sensitive information such as faces, license plates, or documents can be visible in the background. This project provides an automated solution that not only highlights such risks but also offers sanitization options to make photos safe for sharing. By combining awareness through privacy scoring and action through sanitization, the system helps users confidently share images without compromising their security.

### 1.3.2 Scope

### The scope of this project is not limited to just identifying privacy risks but extends to offering practical

### solutions for them. Technically, it makes use of artificial intelligence and computer vision to detect sensitive elements and then assigns a privacy score to indicate the level of risk. It further includes sanitization techniques such as blurring to remove or hide those details. Beyond technical features, the scope also includes building a simple and interactive platform that anyone can use, even without advanced knowledge of editing tools. The project can also be extended to mobile and web platforms, making it versatile and adaptable for different environments.

### 1.3.3 Applicability

### The applicability of the system is very wide. For individuals, it is useful when uploading photos to social

### media, messaging apps, or cloud storage, ensuring that private details are not accidentally shared. For

### organizations, it can be used to secure employee or client information before photos and documents are

### published. Government and corporate portals can integrate this system to check official document uploads,

### while social media companies can adopt it as part of their photo-sharing process. By being adaptable to both personal and professional use cases, the system has the potential to become a standard tool for privacy protection in the digital space.

## 1.4 Achievements

The journey of developing this project on Privacy Scoring and Sanitization of User Uploaded Photos has

resulted in several significant achievements that contribute not only to academic learning but also to solving a real-world problem. One of the foremost achievements of the project is the successful implementation of an AI-driven detection system. Using computer vision and machine learning algorithms, the system can identify sensitive elements such as human faces, vehicle license plates, ID cards, or visible text within an image. This accomplishment is important because it reduces human effort and error, ensuring that sensitive areas are not overlooked during manual checks. The accuracy of detection demonstrates that the project is not just a conceptual idea but a working solution that can be applied in real scenarios.

Another major achievement of this project is the development of a privacy scoring mechanism, which

serves as a clear indicator of risk for users. Most existing image-editing tools allow blurring or cropping, but they do not provide any measurable understanding of how safe or unsafe an image is. With this system, every uploaded photo is assigned a privacy score, making it easier for users to judge whether the image can be shared as it is or needs further sanitization. This scoring mechanism adds a new dimension to privacy protection by creating awareness and offering a straightforward way to interpret potential risks. Users who are not technically skilled can still rely on the score to make informed decisions about sharing images.

The project has also achieved success in creating a sanitization module that automates the editing

process. Instead of depending on users to crop or blur specific regions, the system provides instant

sanitization options like blurring, masking, or pixelating detected sensitive areas. This is a noteworthy

achievement because it directly addresses the biggest challenge faced by most users: the lack of time,

awareness, or skill to edit images carefully. By automating sanitization, the system ensures that photos are

ready to share within seconds, minimizing the chances of accidental data leaks.

Another important achievement is the development of a user-friendly interface that simplifies the entire

process. The system has been designed to allow users to simply upload their image and instantly receive

results, without needing to learn complex tools or settings. From a technical perspective, this achievement

shows the effective integration of backend AI models with a clean and accessible frontend. From a user’s

perspective, it ensures that privacy protection becomes a natural and effortless step in their photo-sharing

routine.

This project has also succeeded in demonstrating the practical applicability of academic knowledge.

Concepts learned in artificial intelligence, computer vision, data security, and web development have been

applied together to build a working prototype. The project therefore represents a significant academic

achievement, as it transforms theoretical learning into a practical tool that can be showcased, tested, and

further improved. It highlights the importance of interdisciplinary skills, as the system combines machine

learning, image processing, database design, and user interface development into one coherent solution.

A further achievement of this project is the demonstration of its scalability and adaptability. While theinitial implementation is designed for individual use, the architecture of the system allows it to be integrated into larger applications. For instance, social media platforms can adopt the system to automatically checkphotos before they are uploaded. Organizations can use it to sanitize official photos or documents, while government portals can implement it to secure sensitive identity uploads. This adaptability ensures that the project is not restricted to academic purposes alone but has potential for real-world impact on a larger scale.

In addition to technical achievements, the project has also succeeded in raising awareness about digital

privacy. During the process of developing and testing the system, it became evident how often people

overlook the risks hidden in everyday images. By providing a clear scoring system and automated solutions, the project encourages users to think about privacy before sharing. This shift in mindset is as important an achievement as the technical system itself, because awareness is the first step toward building safer digital habits.

Finally, this project stands as an achievement in terms of future possibilities and research scope. By

proving that privacy scoring and sanitization can be combined into a single system, it opens doors for further

development in areas like video sanitization, real-time camera monitoring, and mobile app integration.

These possibilities highlight that the project is not only relevant today but also provides a foundation for

more advanced privacy-protection tools in the future.

In summary, the achievements of this project can be seen in multiple dimensions: technically, it has built a

working AI-powered system that detects, scores, and sanitizes images; practically, it provides an easy-to-use

platform that benefits individuals and organizations; academically, it applies knowledge across fields to

create an innovative solution; and socially, it spreads awareness about the importance of privacy in the

digital world. Together, these achievements make the project a meaningful contribution both within and

beyond the academic environment.

## 1.5 Organisation of Report

This project report has been carefully structured to present the entire journey of the system titled “Privacy Scoring and Sanitization of User Uploaded Photos.” The organization of the report follows a logical flow that begins with the basic idea of the project, explains the need and objectives, and then gradually moves through system analysis, design, implementation, and results. Each chapter has been written to serve a specific purpose, ensuring that by the end of the report, the reader has a complete understanding of the motivation, development process, and achievements of the project.

The first chapter, Introduction, lays the foundation by giving an overview of the problem domain. It

explains why privacy in digital images has become a concern in today’s world, how users unknowingly

expose sensitive information through photos, and why existing systems are not sufficient. This chapter also covers the background of the project, its objectives, the purpose it serves, its scope and applicability, as well as the major achievements accomplished during the work. In doing so, the chapter sets the stage for the detailed technical discussion that follows in the later sections.

The second chapter, System Analysis, dives deeper into the study of the problem. It describes the existing methods currently available for image editing or sanitization and highlights their limitations, such as dependency on manual effort and lack of awareness. It then explains the proposed system, which introduces the idea of automated privacy scoring and sanitization, and how it overcomes the drawbacks of the existing approaches. Requirement analysis is also included in this chapter, where both functional and non-functional requirements are defined clearly. This gives the reader an understanding of what exactly the system is expected to achieve and the conditions under which it must operate.

The third chapter, Requirement Analysis, provides more detailed insights into the system’s requirements. It discusses the problem definition in a structured way, lists the specifications in both technical and non-technical terms, and explains the planning and scheduling of the project. The chapter also covers hardware and software requirements and includes conceptual models like ER diagrams, use case diagrams, and activity diagrams. These visual and descriptive models help to understand how different components of the system interact with each other.

The fourth chapter, System Design, focuses on how the project was planned and structured technically. It introduces the major modules of the system such as the upload module, detection module, scoring module, and sanitization module. The chapter also explains data design, database schema, flowcharts, and algorithms used in the system. Additionally, user interface design and security considerations are discussed here to show how the system is not only functional but also safe and user-friendly.

The fifth chapter, Implementation and Testing, covers the practical realization of the project. It explains how the proposed design was converted into actual working code using programming languages,

frameworks, and libraries. This chapter includes details of the tools and technologies used, code snippets,

and the process of integrating different modules together. Testing strategies such as unit testing, integration testing, and beta testing are also described to ensure that the system works as intended and meets all specified requirements.

The sixth chapter, Results and Discussion, presents the outcomes of the project. It provides details of how the system performed when tested with different types of images, the accuracy of detection, and the effectiveness of privacy scoring. User reviews and feedback are also included to highlight the practical value of the system. Discussions in this chapter focus on analysing the strengths of the system, its limitations, and areas that need improvement.

Finally, the seventh chapter, Conclusion, wraps up the project by summarizing the key findings and

contributions. It reflects on how the project successfully addressed the problem of unintentional privacy

leaks in images. The chapter also outlines the limitations faced during development and suggests future

scope, such as extending the system to work on videos, adding real-time monitoring, or integrating mobile

app support. The report ends with references to books, research papers, and online resources that were

consulted throughout the project, giving proper credit to prior work.

Overall, the organization of the report has been designed to take the reader on a smooth journey — from understanding the problem and motivation to seeing the technical implementation and finally appreciating the outcomes. By following this structure, the report ensures that even a reader with limited technical knowledge can understand the importance of the project, while a technical reader can dive deeper into the design and implementation details. This balance makes the report comprehensive, clear, and purposeful.

**CHAPTER 2**

# SYSTEM ANALYSIS

## 2.1 Existing system

### At present, most people who wish to protect their privacy in photos rely on simple, manual editing tools .These tools include cropping unwanted areas, blurring faces or sensitive objects, or covering details with shapes or stickers. For example, someone might use a photo-editing app to blur out their car number plate before posting a picture online. Similarly, scanned documents are often edited manually to hide

### identification numbers or signatures.

### While these methods are widely used, they have a very basic approach to privacy protection. The

### responsibility is entirely on the user to carefully check every detail in the photo and decide what to hide.

### This means that if the user overlooks even one small detail such as text in the background or a reflection in

### glass private information may still be exposed. Moreover, different users have different levels of awareness, so what one person considers safe, another may not even notice.

### In larger contexts, such as organizations, the existing system is also not practical. Employees may be

### instructed to manually sanitize documents or images before uploading them, but this is time-consuming and inconsistent. With the increasing number of images being shared daily, manual methods simply cannot keep up with the demand for speed and accuracy.

### .

### 2.1.1 Limitations of existing systems

The biggest limitation of the existing system is that it depends completely on human effort and judgement .Since users have to manually decide what to edit, the chances of missing sensitive information are very high .This creates a serious risk, because once an image is uploaded, it is almost impossible to fully control where it will be shared or how it will be used.

Another limitation is that manual editing tools are often time-consuming. Even for a single photo, it may take several minutes to carefully identify and blur each sensitive element. For someone who regularly

uploads photos or for organizations dealing with hundreds of images, this approach is not realistic.

There is also the problem of inconsistent results. Two people editing the same photo may hide different

areas depending on their understanding of what is sensitive. This lack of standardization means that privacy cannot be guaranteed. Furthermore, these tools do not provide any indication of how “safe” a photo is. Users may blur certain areas, but they still have no clear measure of whether the photo is now secure enough to share.

Lastly, the existing systems are not designed with automation or scalability in mind. They cannot handle large volumes of data, nor can they be easily integrated into social media platforms, official portals, or corporate systems. This makes them unsuitable for the growing need of fast, consistent, and intelligent

privacy protection.

## 2.2 Proposed systems

## To overcome these challenges, the proposed system introduces a more intelligent and automated solution:

## Privacy Scoring and Sanitization of User Uploaded Photos. Unlike manual methods, this system uses

## artificial intelligence to analyse images automatically and identify sensitive elements such as faces,

## documents, license plates, or background text. By doing so, it removes the dependency on user judgment

## and ensures that nothing important is overlooked.

## The most unique feature of the proposed system is the privacy scoring mechanism. Each uploaded photo is given a score that indicates how safe or risky it is to share. This score acts as a simple and understandable signal for users, even if they are not technically skilled. Instead of guessing whether the photo is safe, they now have a clear measurement to rely on.

## Another important aspect of the system is automated sanitization. Once the sensitive regions are detected, the system gives users options to blur, mask, or pixelate those areas instantly. This eliminates the need for manual editing, saves time, and provides consistent results. Users can download the sanitized version of their photo within seconds, making privacy protection quick and practical.

## In addition to being user-friendly, the system is also scalable and adaptable. It can be integrated into larger platforms such as social media networks, corporate applications, or government portals. This means that privacy checks can become a part of the upload process itself, protecting users even before their photos go online.

## 2.3 Requirement Analysis

The Privacy Scoring and Sanitization system requires a careful balance of **functional** and **non-functional requirements** to ensure it is both effective and user-friendly.

Functionally, the system provides a platform where users can upload their photos and instantly receive a **privacy score** along with a **risk level** indicating how sensitive the image is for online sharing. The system detects sensitive elements within the photo, such as faces, ID cards, license plates, or visible text, and evaluates the risks associated with leaving them unprotected. Once the analysis is complete, the user is presented with the option to apply **sanitization techniques** such as blurring or masking the detected regions. This allows individuals to protect their personal information before sharing the image online. After sanitization, the user can **download the edited photo** directly from the platform. Additionally, the system integrates a **chatbot** that enables users to ask privacy-related questions, receive educational responses, and better understand online safety practices. The overall functionality is designed to provide a simple yet powerful workflow: Upload → Analyse → Score → Sanitize → Download.

In terms of non-functional requirements, the system must offer a high level of **usability**, with a clean and intuitive interface that requires no technical expertise. Users should be able to upload an image and access results in just a few clicks. **Performance** is crucial, as privacy scoring and sanitization should be completed in real time with minimal delays. The system must also ensure **security**, meaning uploaded photos are stored only temporarily during processing and are not saved permanently, thereby protecting user privacy. **Scalability** should be supported so that future versions of the system can include more advanced detection features (like detecting documents, credit card numbers, or sensitive gestures) without redesigning the entire architecture. The platform should be **compatible across devices** such as laptops, tablets, and smartphones, and accessible via major web browsers to maximize reach. Finally, the system must demonstrate **reliability**, handling different image formats and sizes gracefully without errors or crashes.

**Functional Requirements:**

1. **Image Upload Functionality:** The system must allow users to upload an image through a simple, user-friendly interface. The process should support common formats such as JPG, PNG, and JPEG.
2. **Sensitive Data Detection:** Once an image is uploaded, the system should detect sensitive elements such as faces, license plates, ID cards, or text present in the image.
3. **Privacy Score Generation:** The system must calculate a privacy score (ranging from 0–100) based on the detected sensitive elements, and assign a corresponding risk level (Low, Medium, or High).
4. **Result Display:** After analysis, the system should display the uploaded image along with the calculated privacy score and risk level on a results page.
5. **Sanitization Options:** Users should be provided with options such as blurring or masking sensitive parts of the image. Sanitization must be applied automatically without requiring technical expertise.
6. **Image Download:** After sanitization, the system must allow users to download the modified image to their local device for safe sharing.
7. **Privacy Chatbot:** The system should include a chatbot interface where users can ask privacy-related questions. The chatbot should provide meaningful and educational responses to raise awareness of online privacy risks.

**Non-Functional Requirements:**

1. **High Availability and Reliability:** The app should ensure high availability, with uptime approaching 99.9%. Any downtime or interruptions in service could hinder the user experience, especially for those relying on daily tracking or time-sensitive health alerts.
2. **Data Synchronisation:** For users across multiple devices, data synchronisation is key. The app should provide seamless cross-device functionality so that a user can log data on their smartphone and review it later on a tablet or desktop. While keeping local data storage, an optional cloud backup feature could be included to prevent data loss.
3. **Cross-Platform Optimization:** While compatibility across devices is essential, the app should also be optimised for various operating systems (iOS, Android, Windows, macOS) to ensure smooth operation across different platforms without sacrificing performance. Progressive Web App (PWA) technology could be explored to create a unified experience across browsers and devices.
4. **Offline Functionality:** Given that users may not always have internet access, the app should offer offline capabilities, allowing users to log their data and access essential educational content without being connected to the internet. Synchronisation could occur when the connection is restored.
5. **Accessibility Compliance:** The app must comply with accessibility standards, such as the Web Content Accessibility Guidelines (WCAG). This includes features such as voice-over support for visually impaired users, customizable font sizes, and colour contrast settings for users with visual sensitivities.

## 2.4 Hardware Requirements

* **Processor (Local System):** A standard Intel i5 (8th Gen or later) or AMD Ryzen 5 processor is sufficient to run the Flask application and load the pre-trained model. No high-end processor is required since training is already completed in Colab.
* **Memory (RAM – Local):** At least 8 GB RAM is required to handle image uploads and sanitization without lag. For faster performance and multitasking, 16 GB RAM is recommended.
* **Storage Capacity:** A minimum of 20–30 GB of free disk space is required to store the Flask project, the trained model files, and a few test images. Large storage is not necessary since user images are processed temporarily. SSD storage is preferable for quicker file access.
* **GPU (Optional):** For normal use, GPU is not required since the trained model can run on CPU. However, if real-time large-scale processing is needed, an NVIDIA GPU (GTX/RTX series) will improve speed.
* **Input/Output Devices:** A standard keyboard and mouse for development, and a monitor with at least 1080p resolution to clearly view the original and sanitized images.
* **Network Requirements:** While the core model and Flask application can run offline, a stable internet connection is **mandatory for Gemini API integration** (chatbot). The connection must support API requests and responses in real time. A minimum of 10 Mbps speed is recommended for smooth interaction
* **User Devices:** The app should be optimised for performance across different devices and browsers, including Chrome, Firefox, Safari, and mobile platforms.
* **Power Supply:** Reliable power source for uninterrupted development, deployment, and testing phases, ensuring smooth progress in all environments.

## 2.5 Software Requirements

 **Operating System:**

1. The system should be compatible with Windows, Linux, or macOS.
2. For development, Windows 10/11 or Ubuntu Linux is recommended for better package and library support.

 **Backend Framework (Flask):**

1. Flask (Python micro-framework) is required to build the backend of the application.
2. It handles image upload, communication with the trained model, sanitization functions, and integration with the chatbot (Gemini API).

 **Machine Learning Model (YOLOv8):**

1. YOLOv8 is used as the core detection model for identifying sensitive elements in uploaded images (faces, license plates, text, ID cards).
2. The model is pre-trained and downloaded, making it usable offline for inference.

 **Python Libraries:**

1. **OpenCV:** For image processing and manipulation.
2. **Insiteface:** For face detection
3. **Piexif:** To identify exif data
4. **TensorFlow:** Supporting ML model inference.
5. **Ultralytics YOLO:** YOLOv8 implementation for detection tasks.

 **Frontend Technologies:**

1. **HTML, CSS, JavaScript** for building a simple and interactive user interface.
2. HTML handles the structure of pages (upload, result display).
3. CSS ensures styling, responsiveness, and visual appeal.
4. JavaScript provides dynamic interactions (previewing uploaded image, triggering sanitization options).

 **Chatbot Integration (Gemini API):**

1. Gemini API is required to enable a privacy-focused chatbot that answers user queries.
2. This requires a stable internet connection for making API calls and fetching responses in real time.

 **Development Tools:**

1. **Google Colaboratory:** Used for training and testing the YOLOv8 model in a cloud environment with GPU/TPU support.
2. **StarUML:** For creating UML diagrams (ER, Use Case, DFD, Activity, Sequence).
3. **Code Editor (PyCharm):** For local development of the Flask application.

## 2.6 Justification of selection of Technology

The selection of technologies for this project “**Privacy Scoring and Sanitization of User Uploaded Photos**” has been made carefully, ensuring that they match the functional and non-functional requirements of the system. Each technology was chosen to provide simplicity, reliability, and scalability, while keeping the system lightweight and user-friendly.

1. **Flask (Backend Framework):**  
   Flask was selected as the backend framework due to its simplicity, flexibility, and lightweight nature. Unlike heavier frameworks such as Django, Flask provides only the essentials required for building a web application, which makes it an excellent choice for projects with a simple workflow like image upload, analysis, and result display. Flask also integrates seamlessly with Python libraries and machine learning models, allowing smooth execution of the privacy scoring and sanitization tasks.
2. **YOLOv8 (Machine Learning Model):**  
   YOLOv8 (You Only Look Once, version 8) was chosen as the detection model because of its high accuracy and speed in object detection. It is particularly effective in detecting sensitive elements in images such as faces, license plates, and ID documents. Compared to older YOLO versions and other detection models, YOLOv8 offers improved real-time performance, which is essential for providing instant results to users. Its pre-trained weights and ease of integration with Python further support rapid development and deployment.
3. **Python Libraries (OpenCV, Piexif, Pandas,Numpy, TensorFlow):**  
   Python was selected as the core programming language due to its extensive ecosystem of libraries. OpenCV is used for image processing and applying sanitization techniques like blurring and masking. NumPy and Pandas support data handling and computations efficiently. TensorFlow provides deep learning support for YOLOv8, ensuring smooth inference. These libraries are widely used, well-documented, and community-supported, making them reliable for implementation.
4. **Gemini API (Chatbot Integration):**  
   For the chatbot feature, Gemini API was chosen because it offers advanced natural language understanding and provides meaningful, context-aware responses. This helps users ask questions related to privacy and receive educational guidance. Compared to building a chatbot from scratch, Gemini API reduces complexity and ensures accurate, conversational responses, which enhances the user experience.
5. **Frontend Technologies (HTML, CSS, JavaScript):**  
   The frontend was implemented using basic web technologies — HTML for structure, CSS for styling, and JavaScript for interactivity. This stack was chosen because it is lightweight, easy to implement, and compatible with all major browsers and devices. For a project where the focus is on backend processing and ML model integration, a simple yet effective frontend is sufficient.
6. **Google Colaboratory (Model Training):**  
   Google Colaboratory was used for model training due to its free access to GPUs/TPUs, cloud-based execution, and ease of collaboration. Training YOLOv8 locally would require expensive hardware, but Colab provides a powerful environment without the need for such investment. The trained model can then be downloaded and deployed in the Flask application offline, ensuring efficiency and cost-effectiveness.

**CHAPTER 3**

# REQUIREMENT ANALYSIS

Developing the **Privacy Scoring and Sanitization of User Uploaded Photos**system requires a clear understanding of both what the system should do and how it should perform. The purpose of requirement analysis is to define the system’s needs in a structured way so that design and implementation become straightforward. This chapter outlines the functional requirements, non-functional requirements, hardware requirements, and software requirements for the project.

## 3.1 Problem definition

In today’s digital era, photos and images are shared online more frequently than ever before. While this makes communication and self-expression easier, it also increases the risk of exposing private and sensitive information unintentionally. Many individuals upload images without realizing that details such as their faces, ID cards, license plates, or even background text can be misused if they fall into the wrong hands. Once such content is shared on social media or public platforms, it becomes very difficult to control how it is used, leading to risks such as identity theft, and privacy breaches.

Most users lack the technical expertise or tools to analyze whether an image is safe to share. Traditional photo editing tools are either too complex for non-technical users or do not specifically highlight privacy concerns. As a result, individuals continue to share images without being aware of the risks, leaving themselves vulnerable to data misuse.

This project addresses the problem by creating a system that automatically detects sensitive elements in an image, calculates a **privacy score**, and provides a **risk level** that helps users understand how safe their photo is for online sharing. To further support user safety, the system offers **sanitization techniques** such as blurring or masking sensitive parts of the image. Additionally, a **chatbot** is integrated to educate users about privacy-related topics and guide them toward safer online practices.

Thus, the problem being solved is the lack of simple, accessible, and intelligent tools that help users identify and minimize privacy risks before sharing images online.

## 3.2 Requirements Specification

### ****Application Content Selection****

The Privacy Scoring and Sanitization platform lets users upload images and instantly receive a privacy risk assessment. It automatically detects sensitive content such as faces, IDs, documents, location information, or any personally identifiable data. Each image is given a privacy score on a scale of 0–100, helping users understand the level of risk. The platform also offers options to protect privacy by blurring, masking, or removing sensitive areas, all while keeping the image usable and visually clear.

### ****Integration****

The platform is built to work smoothly with other systems. APIs may be available to connect it with content management systems (CMS), social media platforms, or other web applications. This makes it easy for organizations to include automated privacy protection in their workflows without interrupting existing processes.

### ****Progress Tracking****

Users can keep track of all their uploaded images, along with their privacy scores and any sanitization actions applied. The platform uses secure local or server-side storage, allowing users to review past uploads, monitor improvements, and maintain control over their image library over time.

### ****Feedback and Rewards****

As soon as an image is analyzed, users get clear feedback explaining what sensitive elements were detected and how to handle them. Images are rated with a privacy risk level (Low, Medium, or High), and the platform suggests practical ways to make images safer. Optional gamification elements, like badges or scores, may motivate users to consistently improve their privacy awareness and image safety practices.

### ****User-Friendly Interface****

The platform is designed to be simple and engaging. Users can easily upload images through drag-and-drop or file selection, see privacy scores at a glance, and preview any sanitization actions before applying them. Clear visuals, interactive previews, and helpful tooltips guide users through every step, making privacy management intuitive and even enjoyable.

### ****Reporting and Analytics****

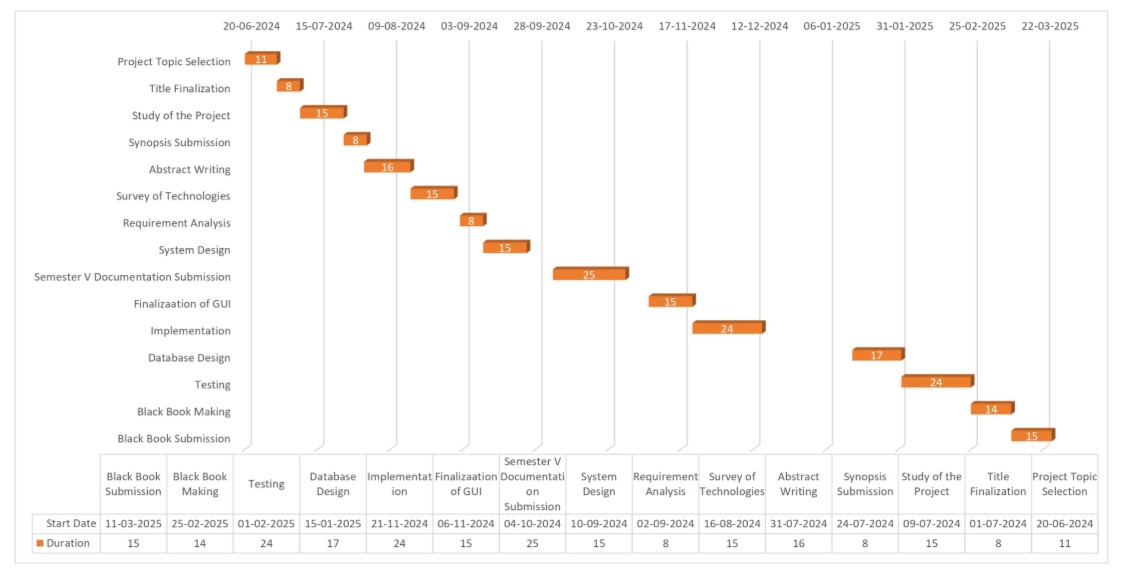
The platform provides useful reports showing image uploads, privacy scores, and applied sanitization actions. Users can see patterns in their images, like frequently detected sensitive content or overall privacy trends. Administrators can access aggregated data to understand usage, assess the effectiveness of privacy measures, and make decisions for future improvements.

## 3.3 Planning and Scheduling

### 3.3.1 Project Planning

|  |  |  |
| --- | --- | --- |
| **Start Date** | **Activity** | **End Date** |
| 20-06-2024 | Project Topic Selection | 30**-**06-2024 |
| 01-07-2024 | Title Finalization | 08-07-2024 |
| 09-07-2024 | Study of the Project | 23-07-2024 |
| 24-07-2024 | Synopsis Submission | 31-07-2024 |
| 31-07-2024 | Abstract Writing | 15-08-2024 |
| 16-08-2024 | Survey of Technologies | 30-08-2024 |
| 02-09-2024 | Requirement Analysis | 09-09-2024 |
| 10-09-2024 | System Design | 24-09-2024 |
| 04-10-2024 | Semester V Documentation Submission | 28-10-2024 |
| 06-11-2024 | Finalisation of GUI | 20-11-2024 |
| 21-11-2024 | Implementation | 14-12-2024 |
| 15-01-2025 | Database Design | 31-01-2025 |
| 01-02-2025 | Testing | 24-02-2025 |
| 25-02-2025 | Black Book MAking | 10-03-2025 |
| 11-03-2025 | Black Book Submission | 25-03-2025 |

### 3.3.2 Project Scheduling



## 3.4 Software and Hardware Requirements

#### 1. Software Requirements

**Operating System:**

* Windows 10/11– compatible with all development tools and libraries used.

**Backend:**

* **Flask (Python micro-framework):** Handles the web server, routing, and processing uploaded images.

**Machine Learning Model:**

* **YOLOv8 (pre-trained):** Integrated for detecting sensitive content in images, such as faces, IDs, and documents.

**Python Libraries:**

* **OpenCV:** For image processing and manipulation.
* **Insiteface:** For face detection
* **Piexif:** To identify exif data
* **TensorFlow:** Supporting ML model inference.
* **Ultralytics YOLO:** YOLOv8 implementation for detection tasks.

**Frontend:**

* **HTML, CSS, JavaScript:** Provides a responsive and interactive interface for uploading images, viewing privacy scores, and applying sanitization.

**Chatbot Integration:**

* **Gemini API:** For answering privacy-related queries within the platform.

**Database / Storage:**

* **Temporary Storage:** static/uploads/ for storing user-uploaded images.

**Development Tools:**

* **Google Colaboratory:** For model training and testing.
* **PyCharm:** Development environment for Python, Flask, and front-end coding.
* **StarUML:** For creating diagrams and planning system architecture.

#### 2. Hardware Requirements

**Development Hardware**

**Processor:**

* Intel i5 (8th Gen) / AMD Ryzen 5 or higher for smooth development and ML inference.

**RAM:**

* Minimum 8 GB (16 GB recommended for faster image processing and multitasking).

**Storage:**

* 20–30 GB free space for project files, models, and temporary image storage.

**GPU (Optional):**

* NVIDIA GTX / RTX series recommended for large-scale or real-time image processing with YOLOv8.

**Input/Output Devices:**

* Standard keyboard, mouse, and monitor (1080p resolution recommended).

**Network:**

* Stable internet connection required for accessing the Gemini API. Local inference can run offline.

### 3. ****Privacy Assessment Modules****

The app will guide users through interactive modules designed to help them understand privacy risks in their images:

* **Sensitive Content Detection:** Users learn how the platform identifies faces, IDs, documents, location data, or other personally identifiable information in their images.
* **Privacy Risk Scoring:** Users can understand how their images are rated on a scale (0–100) based on privacy risk, and what factors contribute to higher or lower scores.
* **Sanitization Options:** Educational guidance on different sanitization methods, such as blurring, masking, or cropping sensitive areas, while keeping images usable and visually clear.

### ****4. Real-World Scenarios****

The platform will simulate practical scenarios where privacy protection is critical:

* **Social Media Sharing:** Users can see how their images might expose sensitive information when shared online and learn how to mitigate risks.
* **Document and ID Handling:** A scenario may simulate uploading IDs or official documents, demonstrating how the app automatically detects and suggests appropriate redactions.
* **Photo Album Privacy:** Users can practice applying sanitization to personal photo collections, learning how to manage privacy consistently across multiple images.

### ****Real-Time Feedback and Progress Tracking****

* Users receive immediate feedback on uploaded images, including detected sensitive content and suggested sanitization actions.
* Progress is tracked locally using secure storage, allowing users to see which images have been scored and sanitized, and to monitor improvement over time.

### ****Accessible on Multiple Devices****

The platform will be accessible on desktops, laptops, tablets, and smartphones, ensuring users can assess and protect their images anytime, anywhere.

### ****Adaptability to User Knowledge Levels****

The app will cater to different levels of user familiarity with privacy:

* **Beginner Users:** Introduced to basic privacy concepts, like detecting faces or IDs in images.
* **Advanced Users:** Explore deeper modules on multiple sensitive content types, advanced sanitization techniques, and batch processing of images.

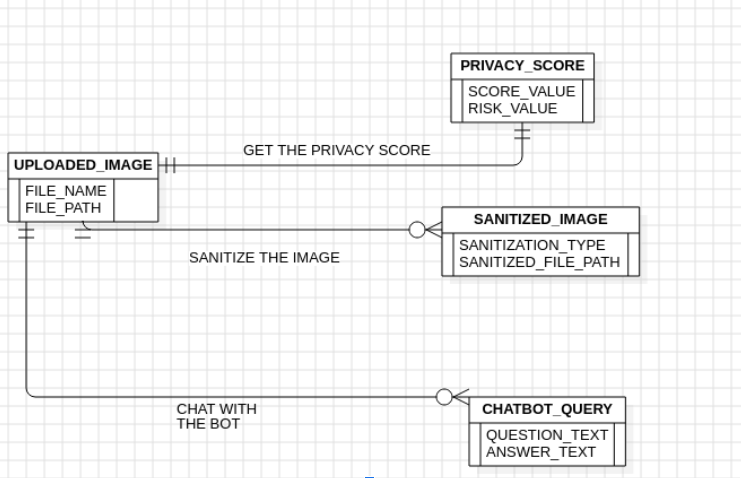
### ****Objective****

By providing an immersive, interactive experience, the app empowers users to understand and manage image privacy effectively. It teaches practical skills, encourages consistent privacy-safe behavior, and fosters confidence in sharing images while minimizing risks of exposing sensitive information.

## 

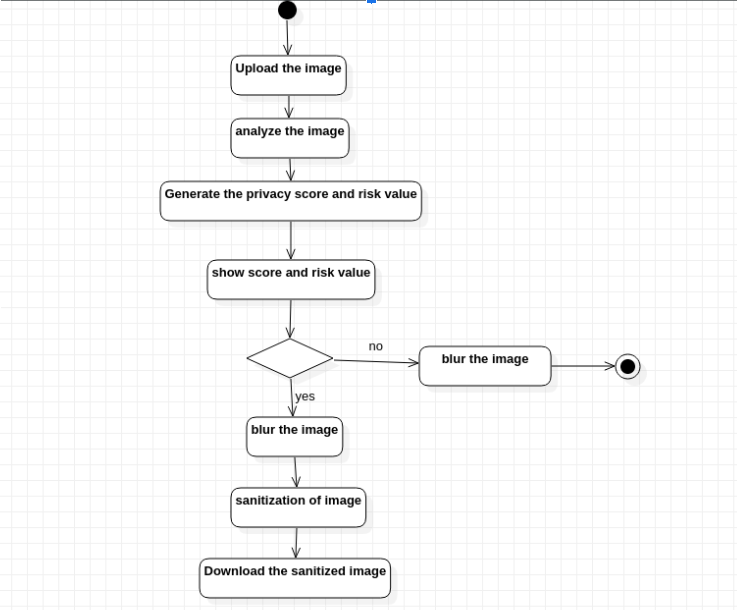
**3.6 Conceptual Models**

### 3.6.1 ER Diagram



The system manages **user-uploaded images**, which are analyzed for **privacy risks**. Each image is assigned a **privacy score** and risk category .Users can **sanitize images** to protect sensitive content, producing a sanitized file. Additionally, users can **interact with a chatbot** to ask questions related to image privacy, storing the query and response.

### 3.6.2 Activity Diagram

****

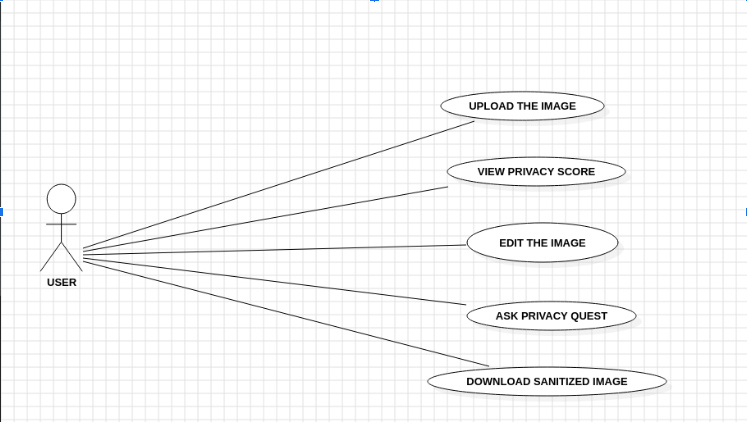
The flowchart outlines a structured process for handling image privacy and sanitization within an application. It begins with the user uploading an image, which is then analyzed to assess potential privacy risks. Based on this analysis, the system generates a privacy score and a corresponding risk value, both of which are displayed to the user. At this point, a decision is made regarding whether the image requires blurring. If blurring is deemed necessary, the image undergoes a blurring process followed by sanitization, after which the sanitized image is made available for download. Interestingly, even if the decision is “no” for blurring, the flowchart still proceeds to blur the image before ending the process—this may be a logical inconsistency worth revisiting. Overall, the flow emphasizes transparency by showing the score before any action is taken, but the conditional logic around blurring could benefit from refinement to align with user expectations and privacy clarity.

### 3.6.3 DFD Diagram

### 

The diagram represents a Data Flow Diagram (DFD) that models the flow of data within an image privacy and sanitization system. It begins with the user uploading an image, which is temporarily stored before being analyzed to generate a privacy score and risk value. These results are then sent back to the user, enabling informed decision-making. In a separate flow, the user initiates a blur or sanitization request, prompting the system to process the image and produce a sanitized version. This sanitized image is then made available for download. The DFD clearly distinguishes between the analysis and sanitization phases, emphasizing modularity and user control. Each component from storage to processing is represented as a distinct entity or process, with directional arrows indicating the movement of data between them.

### 3.6.4 Use Case Diagram



This use case diagram visually represents the interaction between the user and the core functionalities of your privacy-focused image processing system. The user is depicted as an external actor, connected to five distinct use cases: uploading an image, viewing its privacy score, editing the image, asking privacy-related questions, and downloading the sanitized image. Each use case is encapsulated in its own oval, emphasizing modular access to features. The diagram effectively communicates the system’s scope from the user’s perspective, highlighting the available actions without delving into internal logic or data flow. It’s a great way to document functional requirements and clarify what the system offers at a glance.

**CHAPTER 4**

# SYSTEM DESIGN

The system design for IRA, a women's health application, aims to provide an interactive and userfriendly platform for managing period cycles, tracking physical and mental health, and offering educational content. The front-end will be built using ReactTS to ensure responsiveness across devices, with HTML5 and CSS3 for structure and styling. LocalJS will handle data management, allowing users to store progress, symptoms, and quiz results directly on their device, ensuring privacy without the need for logins unless users opt to sync their data. For those who choose data syncing or backups, a minimal backend using Node.js and Firebase can be implemented to manage secure authentication and encrypted data storage. The app’s modular structure will support various health topics such as menstrual health, fitness, and nutrition, with each module featuring interactive lessons, quizzes, and multimedia content. This design ensures scalability, making it easy to add new modules or features over time. Users will also be able to track their symptoms and receive insights into patterns, such as mood swings or stress levels, while quizzes will help them assess their knowledge on reproductive health and wellness topics. Upon completing modules or achieving health goals, users will earn badges and receive real-time feedback to motivate further engagement.

Data privacy is a priority for IRA, with all locally stored information encrypted and any transmitted data secured via SSL/TLS protocols. Compliance with health regulations, such as HIPAA, will ensure user information is handled responsibly. The system will also focus on maintaining data integrity through validation techniques to protect the accuracy of period cycles, health records, and quiz results. With a scalable design and compatibility across mobile and desktop devices, the app can evolve as new health topics or user needs arise. This design not only provides users with a comprehensive health management tool but also ensures a secure, engaging, and educational experience.

## 4.1 Basic Modules

The system for privacy scoring and sanitization of user uploaded photos is divided into a set of basic modules. Each module has a clear role, and together they ensure smooth functioning of the entire application. The modular approach makes the system easier to understand, maintain, and extend in the future. The key modules are described below.

1. **Image Upload Module:**  
   This is the first point of interaction between the user and the system. The user selects an image and uploads it through a simple interface. The image is stored temporarily in the system for further processing.
2. **Sensitive Data Detection Module:**  
   Once the image is uploaded, this module uses the YOLO version eight model to identify sensitive elements such as faces, license plates, or text. It forms the foundation of the project because the quality of detection directly affects the privacy score and sanitization.
3. **Privacy Scoring Module:**  
   Based on the sensitive data detected, the system generates a privacy score. This score lies on a scale from zero to one hundred and helps the user understand how safe it is to share the image. Along with the score, a risk level such as Low, Medium, or High is assigned for easy interpretation.
4. **Sanitization Module:**  
   This module allows the user to protect their image before sharing. Sensitive parts of the image can be blurred or masked automatically. The sanitized image is stored temporarily and made ready for download.
5. **Download Module:**  
   After sanitization, the user can download the edited image. This ensures the user has a safe version of their photo which can be shared online without the risk of exposing sensitive information.
6. **Chatbot Module:**  
   To educate and guide users, a chatbot powered by the Gemini API is integrated into the system. The chatbot answers privacy related questions and provides suggestions to improve online safety.

## 4.2 Data Design

### 4.2.1 Schema Design

## 1. ****Uploaded\_Image Table****

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Data Type | Constraints | Description |
| Image\_ID | Integer | Primary Key, Not Null | Unique identifier for each uploaded image (conceptual only) |
| File\_Name | Varchar(100) | Not Null | Name of the uploaded file |
| File\_Path | Varchar(200) | Not Null | Location of the uploaded file in static/uploads |

### 2. Privacy\_Score

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | Data Type | Constraints | Description |
| Score\_ID | Integer | Primary Key, Not Null | Unique identifier for each privacy score record |
| Image\_ID | Integer | Foreign Key, Not Null | Reference to the related uploaded image |
| Score\_Value | Integer | Foreign Key, Not Null | Reference to the related uploaded image |
| Score\_Value | Integer | Range 0–100, Not Null | Privacy score assigned to the image |
| Risk\_Level | Varchar(10) | Low, Medium, High | Risk category based on score |

### 3. Sanitized\_Image

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** | **Description** |
| Sanitized\_ID | Integer | Primary Key, Not Null | Unique identifier for each sanitized image |
| Image\_ID | Integer | Foreign Key, Not Null | Reference to the related uploaded image |
| Sanitized\_File\_Path | Varchar(200) | Not Null | Location of the sanitized image in static/sanitized |

### 4. Chatbot\_Query

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** | **Description** |
| Query\_ID | Integer | Primary Key, Not Null | Unique identifier for each chatbot interaction |
| Question\_Text | Varchar(250) | Not Null | Question asked by the user |
| Answer\_Text | Varchar(500) | Not Null | Response given by the chatbot |

#### 4.2.2 Data Integrity and Constraints

To ensure the system handles data reliably and securely, several integrity rules and constraints are applied across all tables. These help maintain consistency, prevent errors, and protect the quality of stored information.

**Primary Key Constraints** Each table includes a unique identifier that acts as its primary key. This ensures that every record is distinct and can be accessed without confusion.

* Uploaded\_Image: Image ID uniquely identifies each uploaded image
* Privacy\_Score: Score ID ensures each score record is distinct
* Sanitized\_Image: Sanitized ID tracks each sanitized version separately
* Chatbot\_Query: Query ID identifies each chatbot interaction

**Foreign Key Constraints** Relationships between tables are maintained using foreign keys. These keys link related records and ensure that referenced data exists.

* Privacy\_Score: Image ID links to the corresponding record in Uploaded\_Image
* Sanitized\_Image: Image ID connects to the original image in Uploaded\_Image

**Not Null Constraints** Certain fields must always contain a value to avoid incomplete or invalid records.

* Uploaded\_Image: Image ID, File Name, File Path, and Upload Time must be present
* Privacy\_Score: Score ID, Image ID, Score Value, and Risk Level are required
* Sanitized\_Image: Sanitized ID, Image ID, Sanitization Type, and Sanitized File Path must be filled
* Chatbot\_Query: Query ID, Question Text, Answer Text, and Query Time cannot be empty

**Unique Constraints** Primary keys in each table are inherently unique. This prevents duplication and ensures that each record can be identified independently.

**Check Constraints** Some fields are restricted to specific ranges or values to maintain logical consistency.

* Privacy\_Score: Score Value must be between 0 and 100
* Privacy\_Score: Risk Level must be one of the following — Low, Medium, or High
* Sanitized\_Image: Sanitization Type must be either Blur or Mask

**Default Constraints** Default values are automatically assigned to certain fields when no input is provided.

* Uploaded\_Image: Upload Time defaults to the current timestamp
* Chatbot\_Query: Query Time is set to the current time by default

**Data Validation Techniques** To maintain clean and secure data throughout the system:

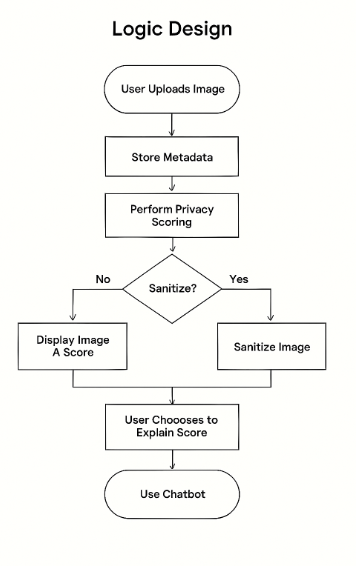
* File names and paths are validated to prevent invalid characters or formats
* Score values and risk levels are checked to ensure they fall within allowed ranges
* Text fields like questions and answers are scanned for length and structure

**Cascade Actions** If an image is deleted from the Uploaded\_Image table, related records in Privacy\_Score and Sanitized\_Image are automatically removed. This prevents orphaned records and keeps the database clean.

**Transaction Management** When multiple updates are needed — such as scoring an image and saving its sanitized version — transactions ensure that all changes happen together. If any part fails, the entire operation is rolled back to maintain consistency.

## 4.3 Procedural Design

### 4.3.2 LOGIC DIAGRAM



The procedural design of the system outlines how different modules work together to provide a seamless experience for users while maintaining data integrity and privacy. The process begins when a user uploads an image through the web interface. This image is received by the backend, stored in a designated folder, and its metadata such as file name, path, and upload time is recorded in the database under the Uploaded\_Image table. A unique Image ID is generated to track this image throughout the system.

Once the image is uploaded, the privacy scoring module is triggered. This module uses AI and computer vision to scan the image for sensitive elements such as faces, documents, or visible text. Based on the findings, a privacy score ranging from 0 to 100 is calculated, and a corresponding risk level Low, Medium, or High is assigned. These values are stored in the Privacy score table and displayed to the user, helping them understand the potential privacy risks associated with the image.

Depending on the score and risk level, the system offers sanitization options to the user. The user can choose to blur to hide sensitive areas. Once selected, the system applies the chosen method and saves the sanitized image in a separate folder. The path are recorded in the Sanitize image table, and the user is given the option to preview or download the sanitized version.

To enhance transparency and user understanding, the system includes a chatbot powered by Gemini. Users can ask questions about their privacy score or the sanitization process. The chatbot processes the query, generates a response, and displays it in the chat interface. Each interaction is logged in the Chatbot\_Query table, including the question, answer, and timestamp.

Throughout these procedures, the system enforces validation checks, maintains foreign key relationships, and uses transaction management to ensure that multi-step operations—such as uploading, scoring, and sanitizing—are completed reliably. Cascade actions are also implemented so that if an image is deleted, all related records in the Privacy\_Score and Sanitized\_Image tables are automatically removed. This design ensures that the system remains consistent, secure, and user-friendly.

### 4.3.2 Data Structures

1. **Uploaded\_Image**

* Image ID: Primary key that uniquely identifies each uploaded image.
* File Name: Name of the uploaded image file.
* File Path: Location of the image stored in the server (e.g., static/uploads).
* Upload Time: Timestamp indicating when the image was uploaded.

1. **Privacy\_Score**

* Score ID: Primary key that uniquely identifies each privacy score record.
* Image ID: Foreign key linking to the Uploaded\_Image table.
* Score Value: Numeric score (0–100) representing the privacy risk level.
* Risk Level: Category based on score (Low, Medium, High).

1. **Sanitized\_Image**

* Sanitized ID: Primary key that uniquely identifies each sanitized image.
* Image ID: Foreign key linking to the Uploaded\_Image table.
* Sanitization Type: Method used to sanitize the image (e.g., Blur, Mask).
* Sanitized File Path: Location of the sanitized image stored in the server (e.g., static/sanitized).

1. **Chatbot\_Query**

* Query ID: Primary key that uniquely identifies each chatbot interaction.
* Question Text: The question asked by the user regarding privacy or score.
* Answer Text: The response generated by the chatbot.
* Query Time: Timestamp indicating when the query was made.

#### Algorithm Design .

Step 1: Image Upload

* Input: Image file from the user.
* Action: Receive file via upload interface.
* Output: Image received and ready for processing.

Step 2: Store Metadata

* Input: Uploaded image.
* Action: Extract and store metadata (e.g., filename, upload time, user ID) in the Images Table.
* Output: Image and metadata saved for reference.

Step 3: Privacy Scoring

* Input: Uploaded image.
* Action: Analyze image for privacy-sensitive content. Assign a privacy risk score using a defined scoring algorithm.
* Output: Privacy score generated for the uploaded image.

Step 4: Sanitization Decision

* Input: Privacy risk score.
* Action: Check if the privacy score exceeds a threshold. If Yes, proceed to sanitize; if No, retain image as-is.
* Output: Decision whether to sanitize the image.

Step 5: Image Sanitization (Conditional)

* Input: Image flagged for sanitization.
* Action: Apply sanitization methods to the image. Update stored image if sanitized.
* Output: Sanitized image saved or original image retained.

Step 6: Display Image Score

* Input: Processed image and privacy score.
* Action: Display privacy score and the relevant image.
* Output: User sees image and privacy assessment.

Step 7: Explain Score (Optional)

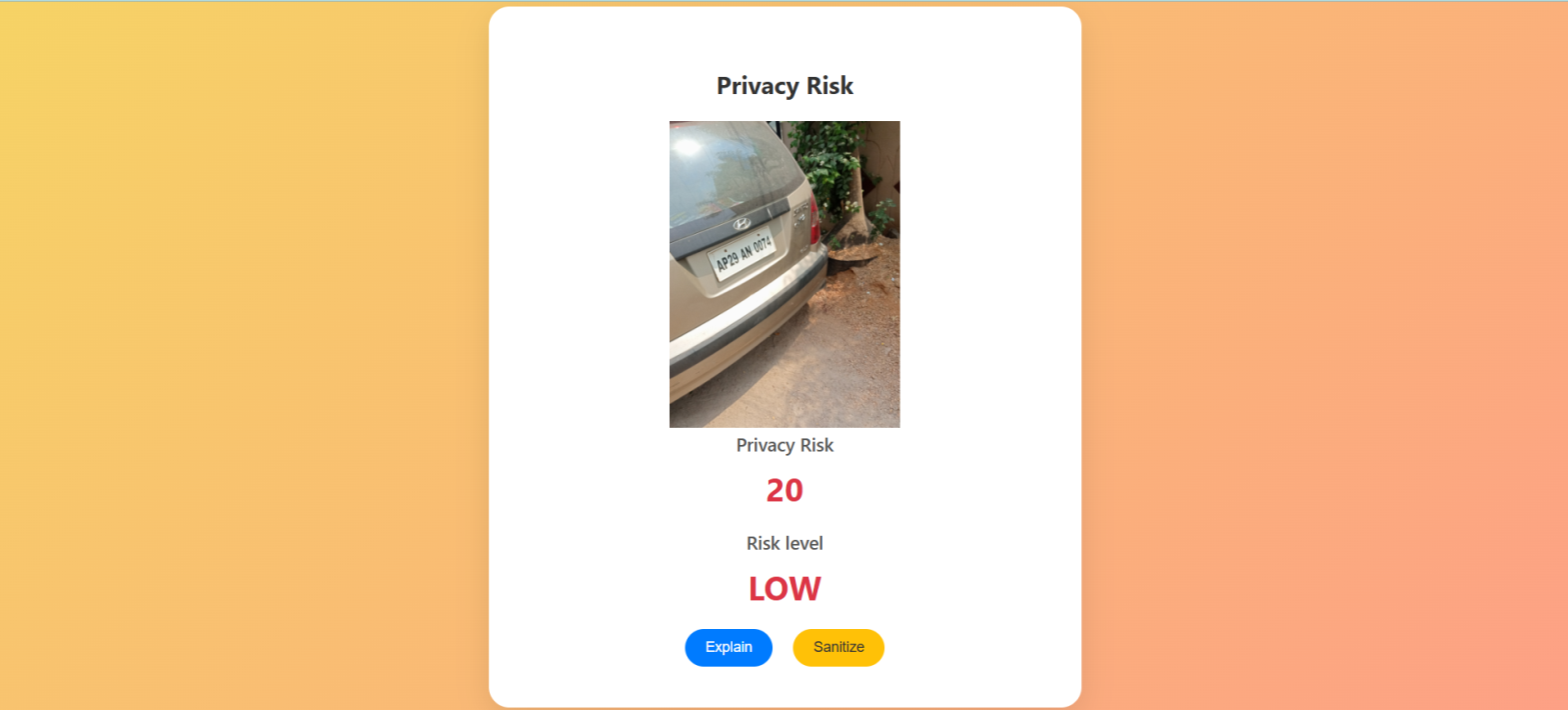
* Input: User action to request explanation.
* Action: Provide option to explain score details.
* Output: Explanation interface is presented.

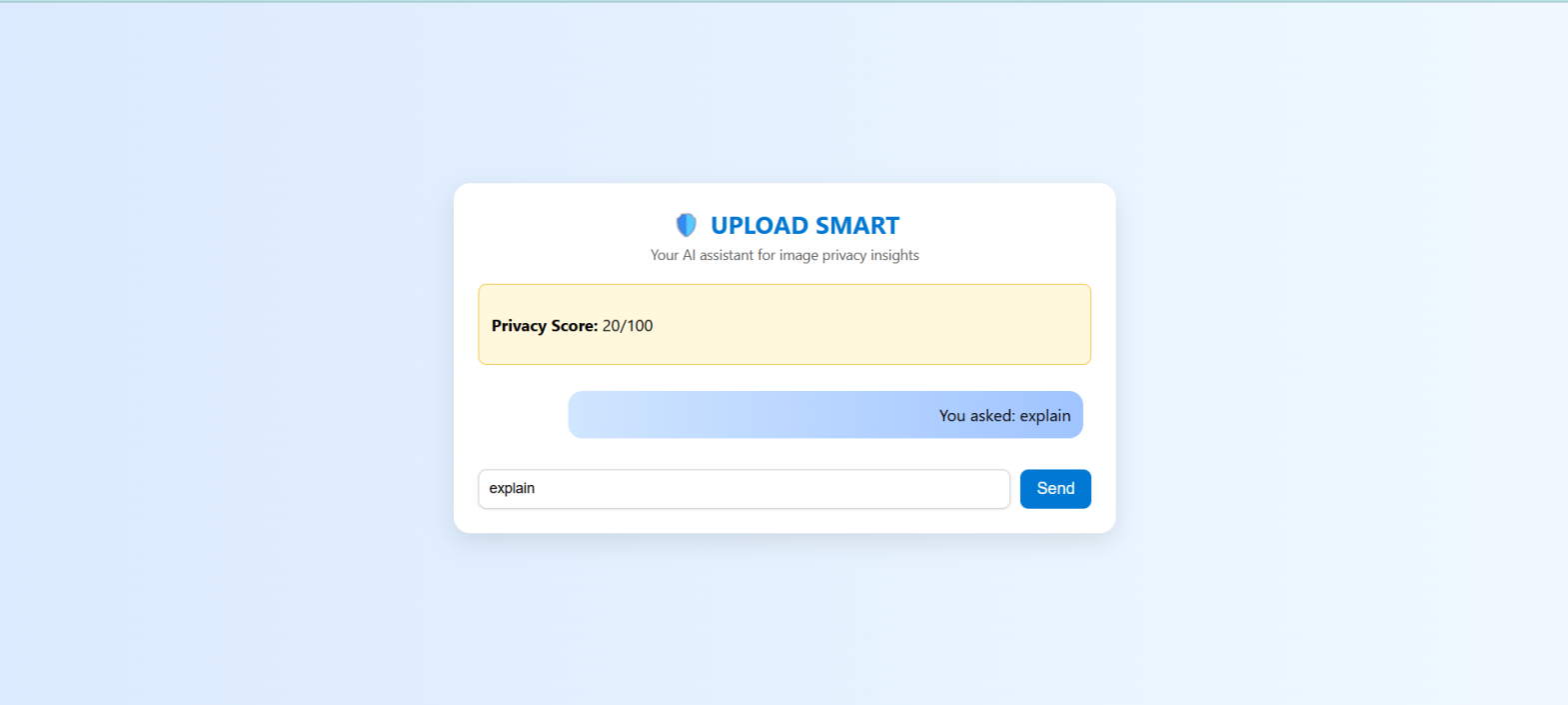
Step 8: Chatbot Assistance

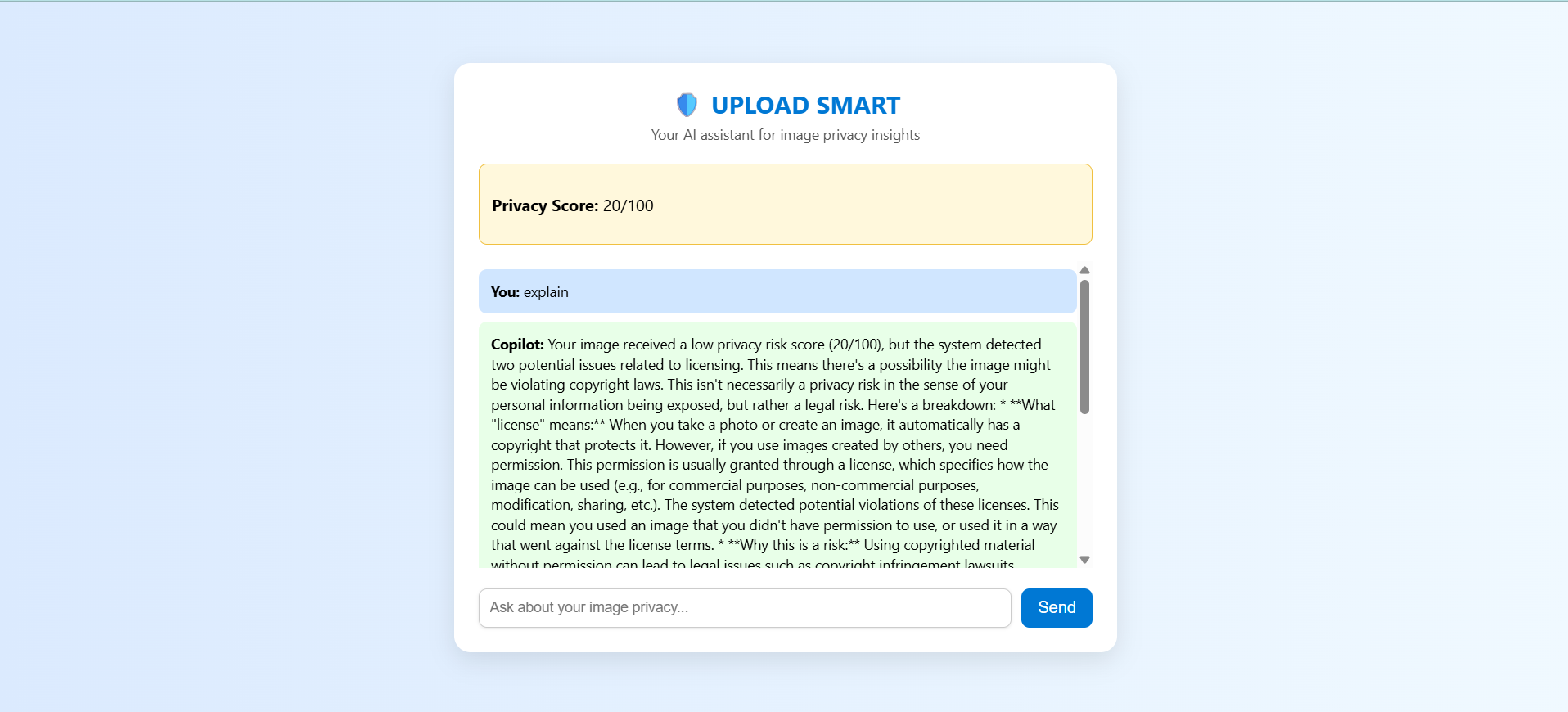
* Input: User requests information or clarification.
* Action: Route user to AI chatbot for explanations and guidance.
* Output: User interacts with chatbot for more details.

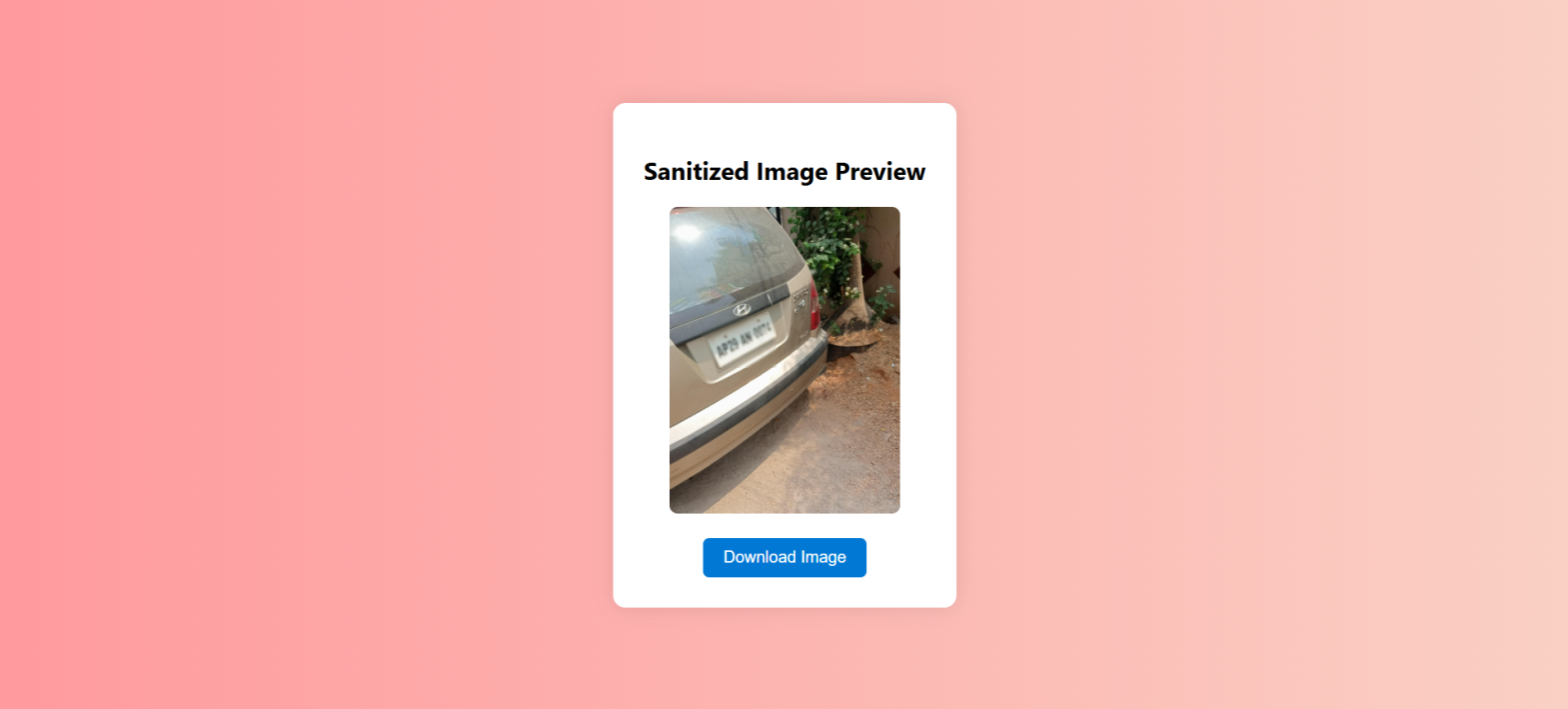
## User Interface Design











1. **Data Privacy**

* **Description**: The system processes sensitive user data such as uploaded images, privacy scores, and chatbot queries. While this data is stored temporarily, protecting it during its short lifecycle is essential for maintaining user trust.
* **Risk**: If temporary files or metadata are mishandled, even briefly, it could lead to unintended exposure or misuse of personal information.
* **Mitigation**:
  + Store all uploaded images in a non-public directory and restrict direct access.
  + Use randomized filenames or hashed IDs to prevent guessable paths.
  + Clearly inform users that their images are processed temporarily and not stored permanently.
  + Avoid logging sensitive data such as file paths or user queries in plain text.
  + Ensure compliance with lightweight privacy standards and obtain user consent where applicable.

1. **Data Integrity**

* **Description**: The system generates privacy scores and sanitization records based on user-uploaded images. Ensuring that this data remains accurate and untampered is important for reliable feedback.
* **Risk**: If privacy scores or sanitization types are altered, users may receive misleading results or lose trust in the system.
* **Mitigation**:
  + Apply strict validation checks during scoring and sanitization.
  + Use hash comparisons or checksum methods to verify data integrity.
  + Maintain audit logs to detect unauthorized changes, even for short-lived data.

1. **Network Security**

* **Description**: The system communicates with external services such as the Gemini chatbot API and may transmit image-related data or user queries. Securing these transmissions is critical to prevent interception.
* **Risk**: Without proper encryption, attackers could intercept sensitive data during transmission, even if it’s not stored permanently.
* **Mitigation**:
  + Use SSL/TLS protocols for all data exchanges.
  + Implement end-to-end encryption for sensitive payloads.
  + Avoid transmitting identifiable metadata unless absolutely necessary.

1. **Access Control**

* **Description**: While the system is user-driven, backend access may be required for managing content, reviewing logs, or integrating third-party services.
* **Risk**: Weak access controls could allow unauthorized users to modify privacy scores, chatbot responses, or system configurations.
* **Mitigation**:
  + Implement role-based access control (RBAC) for backend operations.
  + Use secure authentication methods such as multi-factor authentication (MFA).
  + Log all administrative actions and review them periodically.

## 4.6 Test Cases

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **SR NO.** | **Scenario Description** | **Action to be Performed** | **Expected Values** | **Actual Value** | | 1 | Image Upload | Upload a valid image via the frontend | Image should be saved in static/uploads and metadata stored in database | Image Uploaded Successfully | | 2 | Invalid File Type | Try uploading a non-image file (e.g., PDF, TXT) | System should reject the file and show an error message | Invalid File Rejected Successfully | | 3 | Privacy Score Generation | Upload an image with visible sensitive elements | Score (0–100) and risk level (Low/Medium/High) should be generated | Privacy Score Generated Successfully | | 4 | Blur Sanitization Option | Upload a high-risk image | Blur option should be shown to the user | Blur Option Displayed Successfully | | 5 | Sanitized Image Generation | Apply Blur to the image after scoring | Sanitized image should be saved and linked in the database | Sanitized Image Generated Successfully | | 6 | Chatbot Query Handling | Ask chatbot “Why is my score high?” | Chatbot should respond with relevant explanation and log the interaction | Chatbot Responded Successfully | | 7 | Temporary Storage Cleanup | Wait for scheduled cleanup duration | Uploaded and sanitized images should be deleted from storage | Temporary Files Deleted Successfully | | 8 | Data Privacy Compliance | Review privacy policy and consent flow | User should be informed and consent should be recorded | Privacy Compliance Verified | | 9 | Network Security | Monitor data transmission during upload and chatbot interaction | All transmissions should be encrypted using SSL/TLS | Secure Transmission Verified | | 10 | Cascade Delete Functionality | Delete an image from the database | Related privacy score and sanitized image records should also be deleted | Cascade Delete Verified | |

**CHAPTER 5**

# IMPLEMENTATION AND TESTING

**Chapter Preview**

This chapter outlines the implementation strategy and testing methodology used in the development of the privacy scoring and image sanitization web application. It includes the development approach, technologies used, integration steps, and various testing techniques applied to ensure system reliability, security, and user satisfaction.

### 5.1 Implementation Approaches

## 5.1 Implementation Approaches

The privacy scoring and sanitization system was developed using an incremental and modular approach. This allowed for continuous feedback, iterative improvements, and seamless integration of new features such as chatbot interaction and privacy education. The key implementation phases included:

* **Requirement Analysis** Identified core privacy concerns related to image sharing, user expectations around transparency, and essential features such as privacy scoring, image sanitization, and chatbot-based education. Special attention was given to user flow clarity and minimal data retention.
* **Design** Crafted a clean, gradient-rich interface inspired by Microsoft Copilot aesthetics. The design emphasized intuitive navigation, conversational flow, and visual clarity. Users are guided from image upload to score explanation with minimal friction.
* **Development** Built using Flask for the backend to handle image processing, scoring logic, and database operations. The frontend was developed with HTML, CSS, and JavaScript, styled to reflect a modern, assistant-like experience. Python libraries such as OpenCV and NumPy were used for image analysis and sanitization.
* **Integration** Integrated a Gemini-powered chatbot as a microservice to explain privacy scores and risks. The system also includes temporary image storage, SQLite for lightweight database management, and modular APIs for scoring and sanitization. All components communicate through well-defined routes and shared identifiers.
* **Testing** Multiple testing phases were conducted, including unit testing for scoring logic, integration testing for chatbot responses, and UI testing for upload and preview flows. Security testing focused on temporary storage handling, access control, and encrypted data transmission.

## 5.2 Code

## 5.3 Coding Details and Code Efficiency

**Flask.py**

import google.generativeai as genai

from insightface.app import FaceAnalysis

import cv2

from ultralytics import YOLO

from PIL import Image

import piexif

from flask import Flask, request, render\_template, redirect, url\_for,jsonify,session

import os

from werkzeug.utils import secure\_filename

import requests

import secrets

app = Flask(\_\_name\_\_)

app.config['UPLOAD\_FOLDER'] = os.path.join('static', 'uploads')

os.makedirs(app.config['UPLOAD\_FOLDER'], exist\_ok=True)

genai.configure(api\_key="AIzaSyAhqeBYe7s-64i04iU47ydf8M-6V\_gI9Uk")

model = genai.GenerativeModel("gemini-1.5-flash-latest")

app.secret\_key = secrets.token\_hex(16)

@app.route("/")

def home():

return render\_template("index.html")

@app.route('/privacyscore', methods=['POST'])

def upload\_image():

if 'image' not in request.files:

return "No image uploaded", 400

image = request.files['image']

if image.filename == '':

return "No selected file", 400

filename = secure\_filename(image.filename)

filepath = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

image.save(filepath)

score = privacyapp(filepath)

privacy\_score,\_ = score.privacy\_invade()

privacy\_score,\_ = score.face\_detect()

privacy\_score,\_ = score.show\_gps()

\_,risk\_factor = score.privacy\_invade()

\_,risk\_factor = score.face\_detect()

\_,risk\_factor = score.show\_gps()

risk\_level = ''

if privacy\_score < 40:

risk\_level = 'LOW'

elif privacy\_score < 80:

risk\_level = 'MEDIUM'

else:

risk\_level = 'HIGH'

sanitized\_path = score.blur\_sensitive\_regions()

sanitized\_filename = os.path.basename(sanitized\_path)

session['privacy\_score'] = privacy\_score

session['risk\_factors'] = risk\_factor

session['risk\_level'] = risk\_level

session['sanitized\_filename'] = sanitized\_filename

return render\_template('privacyscore.html', image\_filename=filename, score=privacy\_score, risk\_level=risk\_level)

@app.route('/chat', methods=['GET', 'POST'])

def chat():

privacy\_score = session.get('privacy\_score', 0)

risk\_level=session.get('risk\_level','')

risk\_factors = session.get('risk\_factors', [])

chat\_history = session.get('chat\_history', [])

if request.method == 'POST':

user\_message = request.form['user\_query']

prompt = f"""

A user uploaded an image with a privacy score of {privacy\_score}/100.

and the risk level is {risk\_level}

The detected risks include: {', '.join(risk\_factors)}.

The user asked: "{user\_message}"

Please explain the privacy risks in simple terms and offer suggestions to reduce exposure.

"""

try:

response = model.generate\_content(prompt)

gemini\_reply = response.text

except Exception as e:

gemini\_reply = f"Error: {str(e)}"

# Append to chat history

chat\_history.append({"user": user\_message, "bot": gemini\_reply})

session['chat\_history'] = chat\_history

return render\_template('chatbot.html', score=privacy\_score, chat\_history=chat\_history)

return render\_template('chatbot.html', score=privacy\_score, chat\_history=chat\_history)

@app.route('/preview')

def preview\_blurred():

sanitized\_filename = session.get('sanitized\_filename', 'blurred.jpg')

return render\_template('blur.html', image\_filename=sanitized\_filename)

class privacyapp:

def \_\_init\_\_(self, img):

self.img = img

self.privacy = 0

self.risk\_factors = []

self.blur\_regions = [] # Store regions to blur

def privacy\_invade(self):

model = YOLO('best.pt')

result = model(self.img, conf=0.4)

skip\_class\_id = 7

filtered\_boxes = [box for box in result[0].boxes if int(box.cls.item()) != skip\_class\_id]

result[0].boxes = filtered\_boxes

class\_id = [0, 1, 2, 3, 4, 5, 6]

for box in result[0].boxes:

cls = int(box.cls.item())

if cls in class\_id:

self.privacy += 20

self.risk\_factors.append(model.names[cls])

x1, y1, x2, y2 = [int(v) for v in box.xyxy[0]]

self.blur\_regions.append((x1, y1, x2 - x1, y2 - y1))

return self.privacy, self.risk\_factors

def face\_detect(self):

count = 0

app\_1 = FaceAnalysis(name='buffalo\_l')

app\_1.prepare(ctx\_id=0, det\_size=(1280, 1280))

image = cv2.imread(self.img)

if image.shape[0] > 1000 or image.shape[1] > 1000:

image = cv2.resize(image, (800, 800))

faces = app\_1.get(image)

for face in faces:

count += 1

x1, y1, x2, y2 = [int(v) for v in face.bbox]

self.blur\_regions.append((x1, y1, x2 - x1, y2 - y1))

if count > 0:

self.privacy += count \* 10

self.risk\_factors.append('faces')

else:

print("no faces detected")

return self.privacy, self.risk\_factors

def show\_gps(self):

exif\_dict = piexif.load(self.img)

gps\_data = exif\_dict.get("GPS", {})

if gps\_data:

self.privacy += 20

self.risk\_factors.append('exif\_data')

print('gps data yes')

return self.privacy, self.risk\_factors

def blur\_sensitive\_regions(self, output\_path='static/sanitized/blurred.jpg'):

image = cv2.imread(self.img)

for (x, y, w, h) in self.blur\_regions:

roi = image[y:y+h, x:x+w]

blurred\_roi = cv2.GaussianBlur(roi, (101, 101), 0)

image[y:y+h, x:x+w] = blurred\_roi

temp\_path = "temp\_blur.jpg"

cv2.imwrite(temp\_path, image)

pil\_img = Image.open(temp\_path)

pil\_img.save(output\_path, "jpeg", exif=piexif.dump({}))

os.remove(temp\_path)

print(f"Sanitized image saved at: {output\_path}")

return output\_path

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Privacy Scoring and Sanitization</title>

<style>

body {

margin: 0;

font-family: 'Segoe UI', sans-serif;

background: linear-gradient(135deg, #d4fc79, #96e6a1);

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.upload-container {

background: rgba(255, 255, 255, 0.6);

backdrop-filter: blur(10px);

border-radius: 20px;

box-shadow: 0 8px 30px rgba(0, 0, 0, 0.1);

padding: 40px;

text-align: center;

width: 400px;

transition: all 0.3s ease;

}

.upload-container h3 {

margin-bottom: 20px;

color: #333;

font-weight: 500;

}

.upload-btn {

background: #28a745;

color: white;

border: none;

padding: 12px 24px;

font-size: 16px;

border-radius: 30px;

cursor: pointer;

transition: background 0.3s ease;

}

.upload-btn:hover {

background: #218838;

}

input[type="file"] {

display: none;

}

/\* Modal styles \*/

.modal {

display: none;

position: fixed;

top: 0; left: 0;

width: 100%; height: 100%;

background: rgba(0,0,0,0.4);

justify-content: center;

align-items: center;

z-index: 999;

}

.modal-content {

background: #fff;

padding: 30px;

border-radius: 15px;

text-align: center;

width: 300px;

box-shadow: 0 8px 20px rgba(0,0,0,0.2);

animation: fadeIn 0.3s ease;

}

@keyframes fadeIn {

from { opacity: 0; transform: scale(0.95); }

to { opacity: 1; transform: scale(1); }

}

.modal-content h4 {

margin-bottom: 20px;

font-weight: 500;

}

.modal-buttons button {

margin: 10px;

padding: 10px 20px;

border: none;

border-radius: 25px;

font-size: 14px;

cursor: pointer;

transition: background 0.3s ease;

}

.submit-btn {

background: #007bff;

color: white;

}

.submit-btn:hover {

background: #0056b3;

}

.cancel-btn {

background: #dc3545;

color: white;

}

.cancel-btn:hover {

background: #c82333;

}

</style>

</head>

<body>

<form class="upload-container" id="uploadForm" action="/privacyscore" method="POST" enctype="multipart/form-data">

<h3>Drag & Drop or Click to Add Image</h3>

<label class="upload-btn" for="imageUpload">ADD IMAGE</label>

<input type="file" id="imageUpload" name="image" accept="image/\*" required>

</form>

<!-- Modal -->

<div class="modal" id="uploadModal">

<div class="modal-content">

<h4>Submit this image?</h4>

<div class="modal-buttons">

<button class="submit-btn" onclick="submitUploadForm()">Submit</button>

<button class="cancel-btn" onclick="closeModal()">Cancel</button>

</div>

</div>

</div>

<script>

const imageInput = document.getElementById('imageUpload');

const modal = document.getElementById('uploadModal');

imageInput.addEventListener('change', () => {

if (imageInput.files.length > 0) {

modal.style.display = 'flex';

}

});

function closeModal() {

modal.style.display = 'none';

imageInput.value = '';

}

function submitUploadForm() {

document.getElementById('uploadForm').submit();

}

</script>

</body>

</html>

**privacyscore.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Privacy Score Result</title>

<style>

body {

margin: 0;

font-family: 'Segoe UI', sans-serif;

background: linear-gradient(135deg, #f6d365, #fda085);

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

}

.result-container {

background: #fff;

border-radius: 20px;

box-shadow: 0 10px 30px rgba(0,0,0,0.1);

padding: 40px;

max-width: 500px;

width: 90%;

text-align: center;

}

.result-container h2 {

margin-bottom: 20px;

color: #333;

}

.uploaded-image {

width: 100%;

max-height: 300px;

object-fit: contain;

border-radius: 10px;

margin-bottom: 30px;

box-shadow: 0 4px 12px rgba(0,0,0,0.1);

}

.score-label {

font-size: 18px;

font-weight: 600;

color: #555;

margin-bottom: 10px;

}

.score-value {

font-size: 32px;

font-weight: bold;

color: #dc3545;

margin-bottom: 20px;

}

.action-buttons {

display: flex;

justify-content: center;

gap: 20px;

}

.action-buttons button {

padding: 10px 20px;

border: none;

border-radius: 25px;

font-size: 14px;

cursor: pointer;

transition: background 0.3s ease;

}

.explain-btn {

background: #007bff;

color: white;

}

.explain-btn:hover {

background: #0056b3;

}

.edit-btn {

background: #ffc107;

color: #333;

}

.edit-btn:hover {

background: #e0a800;

}

</style>

</head>

<body>

<div class="result-container">

<h2>Privacy Risk</h2>

<img src="{{ url\_for('static', filename='uploads/' + image\_filename) }}" alt="Uploaded Image" style="max-width: 100%; height: 300px;">

<div class="score-label">Privacy Risk</div>

<div class="score-value">{{ score }}</div>

<div class="score-label">Risk level</div>

<div class="score-value">{{ risk\_level }}</div>

<div class="action-buttons">

<a href="{{ url\_for('chat') }}"><button class="explain-btn">Explain</button></a>

<a href="{{ url\_for('preview\_blurred') }}" ><button class="edit-btn">Sanitize</button></a>

</div>

</div>

</body>

</html>

**chatbot.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Privacy Copilot</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}"/>

</head>

<body>

<div class="gradient-bg">

<div class="chat-container">

<header>

<h1>🛡️ UPLOAD SMART</h1>

<p>Your AI assistant for image privacy insights</p>

</header>

{% if score %}

<div class="score-box">

<p><strong>Privacy Score:</strong> {{ score }}/100</p>

</div>

{% endif %}

<div class="chat-box" id="response">

{% for chat in chat\_history %}

<div class="user-message"><strong>You:</strong> {{ chat.user }}</div>

<div class="bot-message"><strong>Copilot:</strong> {{ chat.bot }}</div>

{% endfor %}

</div>

<form id="chat-form" method="POST" action="/chat">

<input type="text" id="user\_query" name="user\_query" placeholder="Ask about your image privacy..." autocomplete="off" required />

<button type="submit">Send</button>

</form>

</div>

</div>

<script src="{{ url\_for('static', filename='logic.js') }}"></script>

</body>

</html>

**style.css (for chatbot.html)**

body, html {

margin: 0;

padding: 0;

font-family: 'Segoe UI', sans-serif;

background: #f0f4f8;

}

.gradient-bg {

background: linear-gradient(to right, #6a11cb, #2575fc);

min-height: 100vh;

display: flex;

justify-content: center;

align-items: center;

}

.chat-container {

background: white;

border-radius: 16px;

box-shadow: 0 4px 20px rgba(0,0,0,0.2);

width: 90%;

max-width: 600px;

padding: 24px;

}

header {

text-align: center;

margin-bottom: 20px;

color: #333;

}

header h1 {

margin: 0;

font-size: 24px;

}

header p {

margin: 4px 0 0;

font-size: 14px;

color: #666;

}

.chat-box {

max-height: 300px;

overflow-y: auto;

margin-bottom: 20px;

}

.chat-bubble {

padding: 12px;

margin: 10px 8px;

border-radius: 14px;

max-width: 80%;

line-height: 1.4;

}

.user {

background: linear-gradient(to right, #d0e6ff, #a0c4ff);

text-align: right;

margin-left: auto;

}

.bot {

background: linear-gradient(to right, #e0ffe0, #b0f4b0);

text-align: left;

margin-right: auto;

}

/\* Background \*/

.gradient-bg {

background: linear-gradient(to right, #dbeafe, #f0f9ff);

min-height: 100vh;

display: flex;

justify-content: center;

align-items: center;

}

/\* Container \*/

.chat-container {

background: white;

padding: 24px;

border-radius: 16px;

box-shadow: 0 8px 24px rgba(0,0,0,0.1);

max-width: 600px;

width: 100%;

font-family: 'Segoe UI', sans-serif;

}

/\* Header \*/

header h1 {

margin: 0;

font-size: 24px;

color: #0078d4;

}

header p {

margin-top: 4px;

font-size: 14px;

color: #555;

}

/\* Score Box \*/

.score-box {

background: #fff8dc;

border: 1px solid #f0c040;

padding: 12px;

border-radius: 8px;

margin: 16px 0;

}

/\* Chat Box \*/

.chat-box {

margin-bottom: 20px;

}

.user-message, .bot-message {

padding: 12px;

margin: 8px 0;

border-radius: 8px;

font-size: 14px;

line-height: 1.4;

}

.user-message {

background-color: #d0e6ff;

text-align: left;

}

.bot-message {

background-color: #e8ffe8;

text-align: left;

}

/\* Form \*/

form {

display: flex;

gap: 10px;

}

input {

flex: 1;

padding: 10px;

border: 1px solid #ccc;

border-radius: 8px;

font-size: 14px;

box-shadow: 0 1px 3px rgba(0,0,0,0.1);

}

button {

padding: 10px 16px;

background: #0078d4;

color: white;

border: none;

border-radius: 8px;

font-size: 16px;

cursor: pointer;

transition: background 0.3s ease;

}

button:hover {

background: #005fa3;

}

/\* Responsive \*/

@media (max-width: 600px) {

form {

flex-direction: column;

}

button {

width: 100%;

}

}

**logic.js(for chatbot.html)**

document.addEventListener("DOMContentLoaded", function () {

const form = document.getElementById("chat-form");

const responseContainer = document.getElementById("response");

form.addEventListener("submit", async function (e) {

e.preventDefault();

const userQuery = document.getElementById("user\_query").value;

// Show user's question

responseContainer.innerHTML += `

<div class="chat-bubble user">You asked: ${userQuery}</div>

`;

try {

const res = await fetch("/chat", {

method: "POST",

headers: {

"Content-Type": "application/x-www-form-urlencoded",

},

body: new URLSearchParams({

user\_query: userQuery,

}),

});

const data = await res.text(); // Flask returns HTML, not JSON

// Replace entire page with new HTML

document.open();

document.write(data);

document.close();

} catch (error) {

responseContainer.innerHTML += `

<div class="chat-bubble bot error">Error: ${error.message}</div>

`;

}

});

});

**blur.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Blurred Image Preview</title>

<style>

body {

font-family: 'Segoe UI', sans-serif;

background: linear-gradient(to right, #ff9a9e, #fad0c4);

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.container {

text-align: center;

background: white;

padding: 30px;

border-radius: 12px;

box-shadow: 0 0 20px rgba(0,0,0,0.1);

}

img {

max-width: 100%;

height: auto;

margin-bottom: 20px;

border-radius: 8px;

}

button {

padding: 10px 20px;

font-size: 16px;

background-color: #0078D4;

color: white;

border: none;

border-radius: 6px;

cursor: pointer;

}

button:hover {

background-color: #005fa3;

}

</style>

</head>

<body>

<div class="container">

<h2>Sanitized Image Preview</h2>

<img id="blurredImage" src="{{ url\_for('static', filename='sanitized/' + image\_filename) }}" alt="Blurred Image" style="max-width: 100%; height: 300px;">

<br>

<button onclick="downloadImage()">Download Image</button>

</div>

<script>

function downloadImage() {

const image = document.getElementById("blurredImage");

const link = document.createElement("a");

link.href = image.src;

link.download = "sanitized\_image.jpg";

document.body.appendChild(link);

link.click();

document.body.removeChild(link);

}

</script>

</body>

</html>

### 5.3.1 Code Efficiency

To ensure optimal performance and maintainability:

* **Modular Architecture**: Implemented a modular Flask backend with separate routes for image upload, scoring, sanitization, and chatbot interaction to reduce coupling and improve scalability.
* **In-Memory Processing**: Used in-memory image handling with OpenCV to avoid unnecessary disk I/O and speed up sanitization workflows
* .**Temporary Storage Strategy**: Adopted short-lived file storage to reduce data retention risks and simplify cleanup operations..

## 5.4 Testing Approach

A comprehensive testing strategy was adopted to validate the system’s functionality, security, and user experience across all modules.

### 5.4.1 Unit Testing

Individual functions such as image upload, privacy scoring, and blur sanitization were tested using Python’s unit test framework. Each function was validated with both valid and edge-case inputs to ensure predictable behaviour.

### 5.4.2 Integrated Testing

### End-to-end testing was conducted to verify the interaction between modules from image upload to chatbot explanation. Special focus was placed on data flow between the frontend, backend, and database, ensuring consistency and correctness across all components.

### 5.4.3 Beta Testing

A beta version of the application was released to a select group of users. Feedback was collected on UI clarity, chatbot responsiveness, and overall flow. Based on insights, improvements were made to the conversational interface, error handling, and sanitization feedback.

## 5.5 Modifications and Improvements

Based on testing feedback, performance analysis, and evolving user needs, the following enhancements and refinements were made to the system:

* **Sanitization Simplification**: The sanitization module was streamlined to support only the Blur method, ensuring faster processing and a clearer user experience.
* **Conversational Flow Enhancement**: The chatbot’s responses were refined to be more context-aware and user-friendly, helping users better understand their privacy scores and risks.
* **Temporary Storage Optimization**: Auto-deletion scripts were implemented to clean up uploaded and sanitized images after a short duration, reducing storage overhead and enhancing privacy.
* **UI/UX Polishing**: The interface was updated with gradient-rich visuals and Copilot-inspired chat bubbles, improving engagement and clarity across all modules.
* **Error Handling Improvements**: Descriptive error messages were added to guide users during image upload, scoring failures, or chatbot queries.
* **Model Precision and Coverage**: The privacy scoring model is designed to be modular and upgradable. Future improvements will focus on increasing detection precision and expanding the range of identifiable elements (e.g., faces, documents, license plates, handwritten notes), making the system more robust and informative.

## 5.6 Test Cases

Developed detailed test cases covering the following key functionalities of the privacy scoring and image sanitization web application:

* **Image Upload and Metadata Storage** Verified that users can upload images successfully and that metadata such as file name, path, and timestamp is stored correctly in the database.
* **Privacy Score Calculation** Ensured accurate scoring based on image content, with consistent risk categorization (Low, Medium, High) across varied inputs.
* **Blur Sanitization** Tested the blur functionality to confirm that high-risk images are sanitized effectively and saved in the correct location.
* **Chatbot Integration** Validated that the Gemini-powered chatbot responds accurately to user queries and explains privacy scores in a clear, context-aware manner.
* **Temporary Storage Cleanup** Confirmed that uploaded and sanitized images are deleted after a defined duration, ensuring minimal data retention and enhanced privacy.
* **Responsive Design** Tested the application across multiple devices and screen sizes to ensure consistent layout, usability, and accessibility.
* **Security Compliance** Verified that all data transmissions are encrypted and that temporary files are stored securely with restricted access.
* **Cascade Deletion** Ensured that deleting an image also removes its associated privacy score and sanitized version from the database.
* **Error Handling** Checked that invalid uploads, missing scores, or failed chatbot queries trigger appropriate error messages without crashing the system.
* **Model Scalability** Prepared test cases for future model upgrades, including detection of varied sensitive elements such as faces, documents, and handwritten notes.

**CHAPTER 6**

# RESULTS AND DISCUSSION

## 6.1 Test Reports

Testing phases revealed the following key outcomes:

* **High User Engagement:** Beta tester interacted actively with the image upload and chatbot modules, reporting increased awareness of privacy risks and appreciation for the score explanations.
* **System Stability:** The application demonstrated consistent performance across multiple test cycles, with minimal bugs and smooth transitions between modules.
* **Effective Privacy Insights:** Users showed improved understanding of how sensitive elements in images affect privacy scores. The chatbot explanations helped users make informed decisions about sharing or sanitizing their content.
* **Security Compliance:** Temporary storage and encrypted data transmission were validated during testing, confirming that the system meets basic privacy and security standards.

## 6.2 User Documentation

A comprehensive user guide was developed to support onboarding and usage. It includes:

* **Getting Started** Step-by-step instructions on uploading an image, viewing the privacy score, and interacting with the chatbot.
* **Feature Overview** Explanation of privacy scoring, blur-based sanitization, temporary storage handling, and chatbot-based score interpretation.
* **Support Resources** Details on privacy education, FAQs, and contact information for technical assistance or feedback.

**CHAPTER 7**

# CONCLUSIONS

## 7.1 Conclusion

The privacy scoring and image sanitization application successfully empowers users to understand and manage the privacy risks associated with sharing visual content. By providing clear scores, actionable sanitization, and conversational explanations, the system promotes transparency, awareness, and responsible sharing.

### 7.1.1 Significance of the System

 **Privacy Awareness** Helps users identify sensitive elements in their images and understand how these impact their privacy risk.

 **Conversational Clarity** Uses a Gemini-powered chatbot to explain privacy scores in simple, human-friendly language, making technical insights accessible to all users.

 **Modular Scalability** The system’s modular architecture allows for easy updates, integration of new detection models, and expansion of sanitization techniques.

## 7.2 Limitations of the System

 **Detection Scope:** Current models may not detect all types of sensitive content. Only certain elements (e.g., faces, text) are scored, and sanitization is limited to blur.

 **No User Login:** While this simplifies the experience, it limits personalization and long-term tracking of user preferences or history.

 **Device Constraints:** Performance may vary on older devices or browsers with limited support for image processing and dynamic UI rendering.

## 7.3 Future Scope of the Project

To expand the capabilities and impact of the privacy scoring and sanitization system, the following enhancements are proposed:

* **Offline Mobile Application Version:** Develop a lightweight mobile app that functions without internet connectivity, allowing users to assess and sanitize images locally for enhanced privacy and accessibility.
* **Real-Time Mobile Camera Integration:** Enable direct camera access within the app to analyze and score images instantly as they are captured, providing immediate feedback before sharing.
* **Face Replacement:** Introduce advanced sanitization techniques such as face replacement or anonymization, allowing users to retain image context while protecting identity.
* **Offensive Content Detection:** Extend the detection model to identify and flag offensive or inappropriate content, helping users avoid accidental sharing of sensitive or harmful visuals.
* **Social Media Pre-Upload Scanner:** Integrate a scanning module that evaluates images before they are uploaded to social platforms, offering privacy scores and sanitization suggestions in real time.

**7.4 References**

**Related to ChatBots :**

* **Gemini API Reference** – Used for chatbot integration and conversational scoring explanations (internal documentation)

### Technical Documentation & Libraries

* **Flask Documentation** – https://flask.palletsprojects.com
* **OpenCV Python Docs** – https://docs.opencv.org

**Research Papers :**

* Clark, A.; Donahue, J.; Simonyan, K. Efficient Video Generation on Complex *Datasets*. NeurIPS 2020.
* Jocher, G., et al. *YOLO by* Ultralytics*:* Real-Time Object Detection and Segmentation
* piexif Library Documentation: ExifManipulation for Python Images. Official GitHub Repository