Mobile Application for Liveness Checking & NID Text Extraction

Submitted By

MD Zahidul Islam
1949CSE00799
Abdullah Al Mamun Shakib
1949CSE00797
Maidul Hasan
1949CSE00771

Supervisor

Naimul Haque
Assistant Professor

A Project Submitted to the Department of Computer Science and Engineering in Partial Fulfillment of the Requirement for the Degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science and Engineering
MANARAT INTERNATIONAL UNIVERSITY

26 August, 2023

Approval

The project titled, **"Mobile Application for Liveness Checking & NID Text Extraction"** submitted by the following student has been accepted as satisfactory in partial fulfilment of the requirement for the degree of BACHELOR OF SCIENCE in Computer Science and Engineering.

MD Zahidul Islam	**1949CSE00799**
Abdullah Al Mamun Shakib	**1949CSE00797**
Maidul Hasan	**1949CSE00771**

Examination held on 26 August 2023

Approved by

BOARD OF EXAMINERS

Naimul Haque
Assistant Professor
Department of Computer Science and Engineering
Manarat International University

(Supervisor)

Sharmina Zaman
Associate Professor & Head
Department of Computer Science and Engineering
Manarat International University

(**Ex-Officio**)

Mohammad Rofiqul Islam
Associate Professor
Department of Computer Science and Engineering
Manarat International University

(Member)

Md. Zahurul Haque
Assistant Professor
Department of Computer Science and Engineering (Member)
Manarat International University

Tanvir Ahmed
Assistant Professor
Department of Computer Science and Engineering (Member)
Manarat International University

Jakia Akhter
Lecturer
Department of Computer Science and Engineering (Member)
Manarat International University

Shreshtha Sayantika Maitra
Lecturer
Department of Computer Science and Engineering (Member)
Manarat International University

Dr. Md. Ezharul Islam
Professor
Department of Computer Science and Engineering (External Member)
Jahangirnagar University, Dhaka

Declaration

We declare that this project report is our work and has not been submitted for any other degree or professional qualifications. All sections of the paper that use quotes or describe an argument or concept developed by another author have been referenced in the reference section.

26 August, 2023

Name	Signature
MD Zahidul Islam ID: 1949CSE00799	
Abdullah Al Mamun Shakib ID: 1949CSE00797	
Maidul Hasan ID: 1949CSE00771	

Acknowledgement

First, I show my deepest gratefulness to Almighty Allah for giving His blessings on me and giving me the ability to complete this work successfully.

I just wanted to take a moment to express my gratitude and appreciation to everyone who helped make this project a success. Without tireless collaboration and continual direction from those involved, we wouldn't be where we are today. Words can hardly express how thankful I am for all of your hard work and dedication.

Of course, a special thanks goes out to our supervisor, **Naimul Haque**. Your oversight, direction, support, and inspiration were invaluable during the implementation phase. Thank you for enabling us to make wise decisions and for always having our best interests at heart. Your efforts are truly appreciated and will not be forgotten.

We are also thankful to our senior brothers for helping us in the study of this project by giving their valuable suggestions and helping hands.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

26 August, 2023

Abstract

Innovative solutions are required to improve security and improve user experiences since identity verification and document processing are becoming more and more dependent on mobile devices. In response to this requirement, we provide a mobile application created for the iOS operating system that is intended to carry out liveness verification and text extraction for National ID (NID). This application uses state-of-the-art face recognition technology to guarantee liveness in authentication procedures, reducing the dangers associated with submissions of static images. It also uses optical character recognition (OCR) technology to extract text information from NID documents, facilitating quick and precise data entry. The strength of iOS programming is combined with strong security safeguards and effective data processing in our project. Individuals may quickly authenticate their identity and retrieve pertinent information from their NID papers using a simple user interface. Numerous industries, including financial services, e-government programs, and identity verification systems, find use for this application. The methodology section offers details on the technical features of our application as we dive into the specifics of our development procedure. The efficacy and dependability of our liveness checking and NID text extraction functions are shown by preliminary findings. As a result, our mobile application provides a thorough remedy for strengthening identity verification procedures on iOS devices while also guaranteeing customer comfort and security. Its potential effects might be seen across a range of businesses, answering the growing demand for thorough identity verification in a world that is becoming more digital.

Table of Contents

List of Figure

Chapter 1

1 . INTRODUCTION

Identity verification must be secure and effective in our constantly changing digital environment. Our research, "Liveness Checking and NID Text Extraction," is at the center of this technological confluence that has resulted in unique methods for identity identification. This project has two goals: to redefine identity verification through in-process liveness evaluation and to accelerate the procedure by precisely extracting data from National ID (NID) documents.

Liveness Checking: A New Frontier in Identity Verification

Traditional identity verification techniques are no longer as effective in the digital era due to new dangers. The first part of our initiative, "Liveness Checking," forays into uncharted territory by putting a focus on in-the-moment communication during authentication. Approaches relying on static images in the past are susceptible to deception. Our system, in contrast, relies on dynamic evaluations to guarantee the presence of a living person, strengthening the security of identity verification procedures.

NID Text Extraction: Enhancing Efficiency

The second facet of our project, "NID Text Extraction," is also transformational. Long a source of mistakes and inefficiency, manual data entering from NID documents. In this article, we provide cutting-edge Optical Character Recognition (OCR) technique to quickly and precisely extract text from NID documents. This development represents a big step towards more effective document processing in identity verification since it not only improves data correctness but also speeds up the authentication process.

1.1 Background

i. Early Research (2010s):

Researchers started looking at liveness detection as a crucial part of biometric authentication in the early 2010s. To discriminate between real and fake face photos, a

number of approaches were studied, including texture analysis, motion detection, and depth sensing.

ii. Integration with Mobile Devices (Mid-2010s):
Researchers and developers began focusing on including liveness checks into mobile apps as cellphones proliferated. The benefit of front-facing cameras and computing capacity for real-time analysis was provided by mobile devices.

iii. Deep Learning Revolution (Late 2010s):
Deep learning methods, notably convolutional neural networks (CNNs), have been more popular for liveness testing as of late 2010. These deep learning models made it possible to identify facial movements and other indicators of liveness with greater accuracy and robustness.

iv. SDKs and APIs (Late 2010s - Early 2020s):
Software development kits (SDKs) and APIs were made available by businesses and organizations, making it simpler for developers to include liveness checks into iOS apps. These technologies frequently have capabilities like head movement analysis, eye blink recognition, and anti-spoofing safeguards.

v. Privacy Concerns and Regulations (Ongoing):
Concerns regarding user privacy and data security increased as liveness checking technologies developed. Biometric data collection and usage are restricted by laws and regulations including the General Data Protection Regulation (GDPR) of the European Union and other national privacy laws.

vi. COVID-19 Pandemic (2020s):
Touchless and secure identification techniques have become more popular as a result of the COVID-19 epidemic. In situations where physical touch was banned, such mobile payments and access control systems, liveness checking became more important.

vii. Continuous Improvement (Ongoing):
concentrated on enhancing precision, robustness, and the capacity to recognize sophisticated spoofing attempts such as 3D-printed masks or deep-fake movies.

1.2 Problem Statement

Identity verification is crucial for safe and trustworthy transactions, access, and services across many industries, including banking, e-government, and online platforms, in a world that is becoming more and more digital. Traditional identity authentication techniques, however, have a number of serious difficulties:

a. Static and Vulnerable Verification Methods: Traditional identity verification methods that rely on static image-based verification are inherently prone to fraud. By offering static pictures or duplicates, fraudsters can simply get around such techniques.
b. Inefficient Data Entry: During identity verification, manual data entry from National ID (NID) papers is prone to mistakes and inefficiencies. These procedures' precision is highly dependent on the human operator, which causes irregularities and delays.

c. User Experience: Existing identity verification methods frequently fall short of user expectations in terms of user experience. Users are not only irritated by lengthy procedures, delays, and verification problems, but they are also discouraged from utilizing digital services.

d. Security Concerns: Cyberattacks are becoming more sophisticated and more frequent as the digital environment changes. Significant concerns include identity theft, account takeovers, and other fraudulent actions. The security and reliability of identification verification procedures must be guaranteed.

These issues are addressed head-on in our project, "Liveness Checking and NID Text Extraction," created for the iOS platform. We want to reimagine identity verification and document processing by utilizing iOS development's capabilities. In especially for mobile devices, our research aims to improve the security, effectiveness, and overall user experience of identity verification using real-time liveness checks and sophisticated Optical Character Recognition (OCR) algorithms.

1.3 Aims and Objectives

Aim:

Our project's main goal is to create a cutting-edge iOS mobile application that solves the problems with identity verification and document processing. This program is intended to increase security, improve productivity, and offer a seamless user experience

Objectives:

Liveness Checking:

i. **Real-Time Liveness Checking**: Create a reliable liveness checking module that can precisely determine a person's liveness in the course of the identification verification procedure.

ii. **Preventing Spoofing and Impersonation:** The goal of liveness checking is to make sure that the individual requesting to authenticate is actually there and not posing as himself via a still picture, video, or other means. As a result, spoofing attacks—in which an attacker attempts to acquire access by posing as a non-living entity—are reduced.

iii. **Reducing False Positives**: Reducing false positives in biometric authentication systems may be achieved via liveness checks. Liveness checks can assist in distinguishing between genuine, alive users and impersonation efforts. Without them, certain biometric systems might incorrectly verify a user using a high-quality image or a recorded voice.

iv. **Security Enhancement**: The main goal of adding liveness checking to an iOS app is to increase security by making sure that the person trying authentication is actually there and isn't using faked or pre-recorded information. This lessens the likelihood that sensitive data or services may be accessed improperly.

v. **Preventing Identity Theft:** A vital instrument in the battle against fraud and identity theft is liveness checking. It helps protect people's identities and lowers the possibility of illegal access to their accounts and personal information by verifying that the user is physically present.

vi. **Authentication Confidence**: By confirming that the individual being verified is physically present and actively taking part in the verification, liveness checks seek to increase the trustworthiness of the authentication process.

vii. **Adaptability**: Checking for liveness should function on multiple iOS devices and adjust to diverse user circumstances and environmental conditions.

viii. **User Privacy:** Liveness checks have the key goal of protecting user privacy by not requiring the storage or transfer of private biometric data while maintaining high security..

NID Text Extraction:

i. **Data Accuracy:** The main objective of NID text extraction is to precisely extract data from papers or National ID cards. This involves gathering information such as name, birthdate, ID number, and other pertinent facts.

ii. **Efficiency:** Data entry may be streamlined and human input mistakes can be decreased with NID text extraction. By automatically filling up forms or profiles with data from their ID cards, it can save consumers time.

iii. **Error Reduction**: The initiative aims to eliminate manual data entry mistakes that might happen when users manually enter their information by automating the extraction of NID data. This results in the app's data being more accurate and trustworthy.

iv. **Security:** A primary priority is maintaining security. The NID data that was collected has to be treated carefully, not just kept. Ensuring that user data is shielded from unwanted access or breaches requires adherence to data protection and privacy rules.

v. **Integration**: The workflow of the app should incorporate the collected NID data with no issues. This means that programmers should create the system in a way that allows the extracted data to be utilized for a variety of tasks, such pre-filling forms, making user profiles, or doing identification checks.

vi. **Scalability:** The NID text extraction method should be scalable to handle a high amount of queries effectively if the app is anticipated to have a big user base, preventing it from becoming a bottleneck.

vii. **Compliance**: NID text extraction makes ensuring that the retrieved data is trustworthy and complies with official standards if the project calls for validating user IDs or adhering to legal requirements.

viii. **User Education:** Teaching people how to utilize the NID text extraction capability efficiently is one of the goals. Users should be given clear instructions and advice to make sure they comprehend the procedure and its advantages.

By achieving these objectives, our project aims to provide a cutting-edge mobile solution that revolutionizes identity verification and document processing, offering a more secure, efficient, and user-centric experience for individuals and organizations alike.

Chapter 2

Project Requirements Overview

2.1 Platform and Tools:

- **Requirements:**

Certainly, creating an iOS-based mobile application for "Liveness Checking and NID (National ID) Text Extraction" using tools such as Xcode, MLKit, UIKit, and the Swift programming language involves several key steps and components.

Platform: iOS

Development Tools: Xcode, Swift

Machine Learning Framework: MLKit

User Interface (UI): Design the user interface using UIKit components. Create screens for user interactions, including the main authentication screen and any additional screens for settings or user profiles.

What is XCode:

XCode is an integrated development environment (IDE) designed by Apple for macOS, iOS, watch OS, and TV OS app development. It provides a comprehensive set of tools and features that make it easier for developers to create, test, and distribute applications for Apple's various platforms.

Swift:

Swift is a powerful and versatile programming language developed by Apple Inc. It was introduced in 2014 and is designed primarily for building applications for Apple's ecosystem, including iOS, macOS, watch OS, and TV OS. Swift was created to be a modern, fast, safe, and easy-to-learn language for software development.

MLKit:

ML Kit, developed by Google, is a powerful mobile machine-learning framework that allows developers to integrate machine-learning capabilities into their Android and iOS applications with ease. ML Kit simplifies the process of incorporating machine learning models for tasks

like image recognition, text recognition, face detection, barcode scanning, and more, without requiring a deep understanding of machine learning or computer vision.

Why use the iOS Platform?

Choosing the iOS platform for a "Mobile Application for Liveness Checking and NID (National ID) Text Extraction" project involves a series of considerations that stem from the platform's strengths and suitability for specific use cases. Below, I'll detail the advantages of using the iOS platform for this project:

i. **Robust Ecosystem**:
   a. Stability: iOS is known for its stability and consistency across devices, which simplifies development and testing efforts.
   b. Regular Updates: Apple regularly releases updates for its iOS operating system, ensuring that the app remains compatible with the latest devices and features.

ii. **Security and Privacy**:
   a. Data Protection: iOS offers robust data protection features, helping ensure the security and privacy of user data, which is crucial when handling sensitive information like NID data.
   b. App Sandbox: Apps on iOS run in a sandboxed environment, limiting their access to system resources and enhancing security.

iii. **Performance:**
   a. Optimized Hardware: iOS devices are known for their high-performance hardware, which can provide a smooth and responsive user experience.

   b. Swift Programming Language: Swift, Apple's preferred language for iOS development, is designed for performance, making it well-suited for tasks like liveness checking and text extraction.

iv. **User Base and Market Reach:**
   a. Large User Base: The iOS platform boasts a substantial user base, which can be advantageous for reaching a broad audience, especially if your project targets users who predominantly use iOS devices.

   b. High-Value Users: iOS users often have higher purchasing power and are more likely to pay for premium apps or services, potentially benefiting monetization efforts.

v. **Developer-Friendly Tools:**
   a. ML Kit: While primarily associated with Android, Google's ML Kit can be used on iOS, making it easier to incorporate machine learning-based features like liveness checking and text extraction into your app.

   b. UIKit: iOS offers UIKit, a versatile framework for creating interactive user interfaces.

Chapter 3

Design

3.1 NID Text Extraction:

Three diagrams were mentioned for this project. which are,

1.Use case diagram
2.Entity relationship diagram and
3.UML diagram.
First, let's discuss use case diagrams.

3.1.1 Use case Diagram:

An iOS smartphone user is required for this. The user in this diagram scans an NID Card using the camera on an iPhone. User must use an iOS device to scan their NID card.

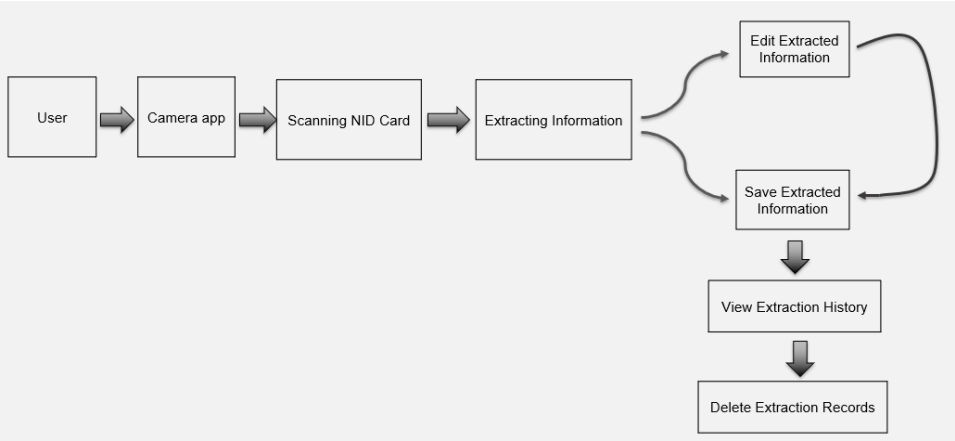

Figure 3.1 : NID Text Extraction Use Case Diagram

After scanning, if the NID card is legitimate, data will be extracted from it. It can retrieve the name, birthdate, and ID card number of the cardholder. We may check the history of the extraction and remove the extraction records if we scan several cards from one device.

3.1.2 Entity Relationship Diagram:

In entity relationship we can see we have some basic relationship with entities. We have some basic relationship between User, NID card, image and extraction.

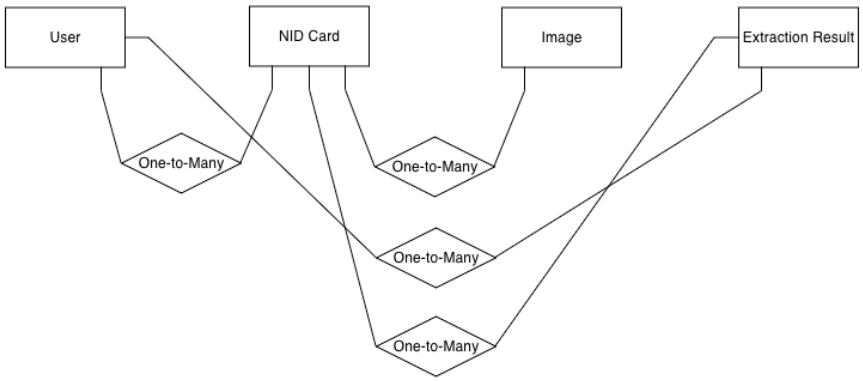

Figure 3.2: NID Text Extraction Entity Relationship Diagram

A user can have multiple NID card but a NID card belongs to one user. Same as a NID card can have multiple images but An image belongs to one NID card. A user can have multiple extraction but an extraction result belongs to one user. A NID card can have multiple extraction results but an extraction result belongs to one NID card.

3.1.3 UML Diagram:

- **Class Diagram:**

A Class Diagram represents the main classes and their relationships in the app.

AppDelegate: Manages app lifecycle and initial setup.

ViewController: Represents the primary user interface of the app.

CameraManager: A class for handling device camera functionality.

TextExtractor: Handles text extraction from images

ResultView: Displays the extracted text information.

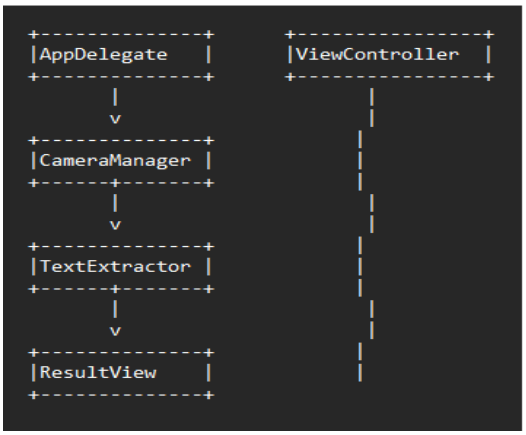

Figure 3.3: NID Text Extraction UML Class Diagram

- **Sequence Diagram:**

A Sequence Diagram illustrates the interactions between objects in a specific scenario, such as a user capturing an image and extracting text from it:

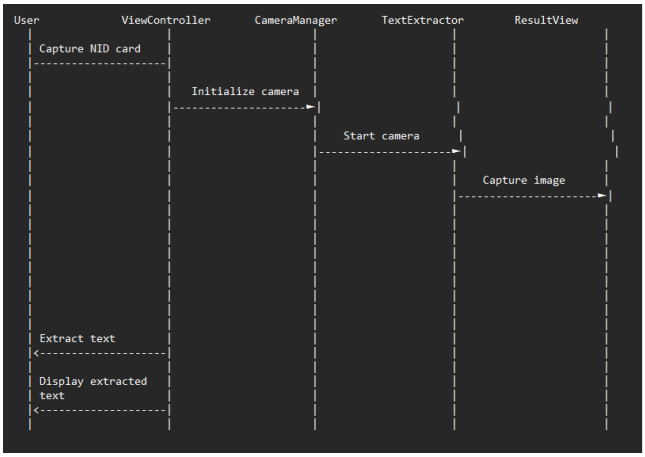

Figure 3.4: NID Text Extraction UML Sequence Diagram

3.2 Liveness Checking:

There are three diagrams. Which are;

1.Use case diagram
2.Entity relationship diagram
and 3.UML diagram.

3.2.1 Use case diagram:

We are integrating some basic functions Because this project is in the initial phase. First of all, users will open this app, Then the app will pop up a front camera preview, Which will detect the user's face , And it will check if this is a live face or a photo. Then the app will show some instructions at the top of the camera preview.

.

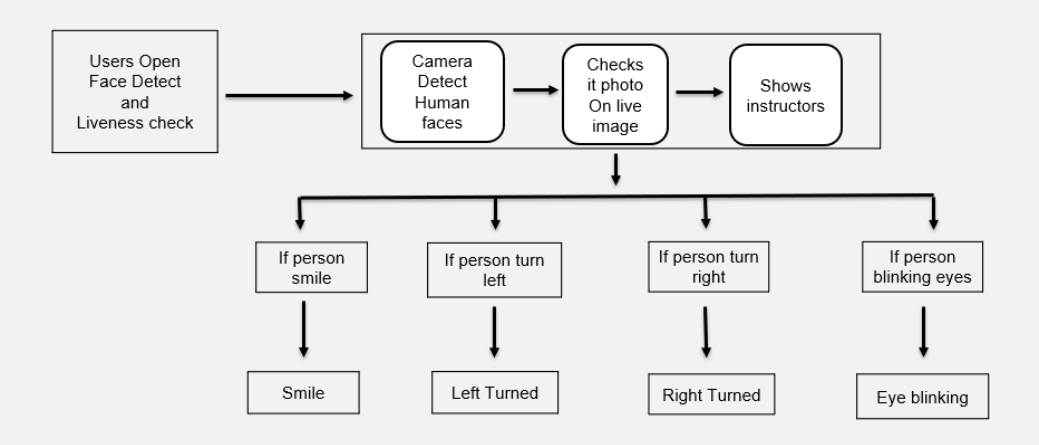

Figure 3.5: Liveness Checking Use Case Diagram

We integrated some functions, There are, If the person smiles, then the XCode output will show a smile, If the person turns left, then it will show left turned, if the person turns right then it will show right turned, if the person blinks his eyes, then it will show eye blinking.

3.2.2 Entity Relationship Diagram

As we can see this is in an initial phase that's why we have some basic relationships with entities.
When the user will enter the app, then the app will create a unique IP key to recognize this user.

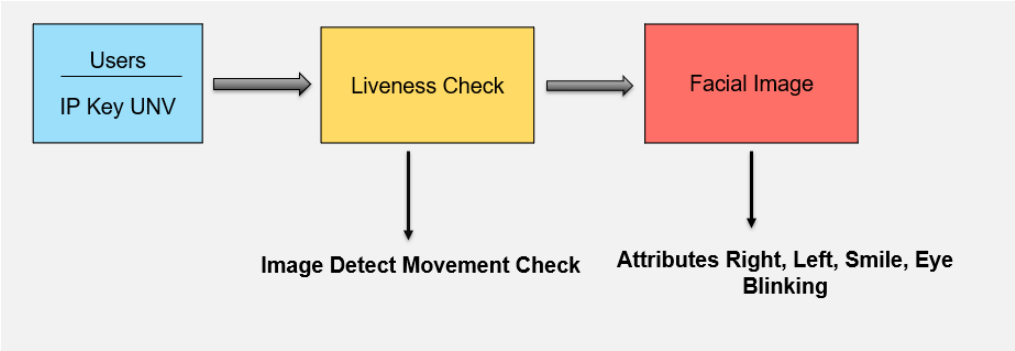

Figure 3.6: Liveness Checking Entity Relationship Diagram

Then we have a liveness check entity, which will detect the movement of the user. It will simply check that the user is using a photo or live image. It will just CHECK. Then the facial image will track the user's facial activities.

3.2.3 UML Diagram

Firstly, the user captures their image through the camera manager, then the face detector detects their face, then the liveness checker comes and checks the liveness of the collected image or face. after doing all this process, then it finally shows the results.

- **UML Use Case Diagram:**

A Use Case Diagram outlines the interactions between the app and its users.

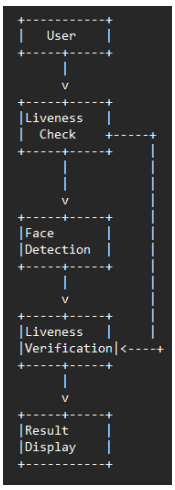

Figure 3.7: Liveness Checking UML Use Case Diagram

User: Represents the end user interacting with the app.

Liveness Check: The main use case representing the core functionality of the app.

Face Detection: An extension of the Liveness Check use case, representing the detection of the user's face.

Liveness Verification: Another extension use case representing the actual liveness verification process.

Result Display: Shows how the app displays the verification result to the user.

- ## Class Diagram:

A Class Diagram represents the main classes and their relationships in the app.

AppDelegate: Manages app lifecycle and initial setup.

ViewController: Represents the primary user interface of the app.

CameraManager: A class for handling device camera functionality.

FaceDetection: Handles face detection functionality.

LivenessVerifier: Performs liveness verification.

ResultView: Displays the verification result.

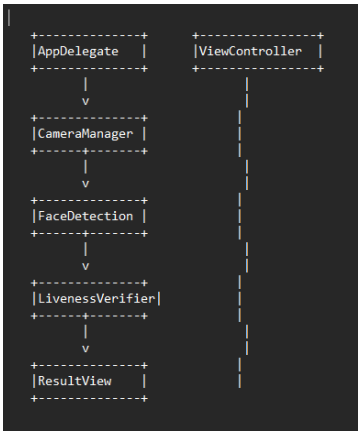

Figure 3.8: Liveness Checking UML Class Diagram

- **Sequence Diagram:**

A Sequence Diagram illustrates the interactions between objects in a specific scenario, such as a user performing a liveness check:

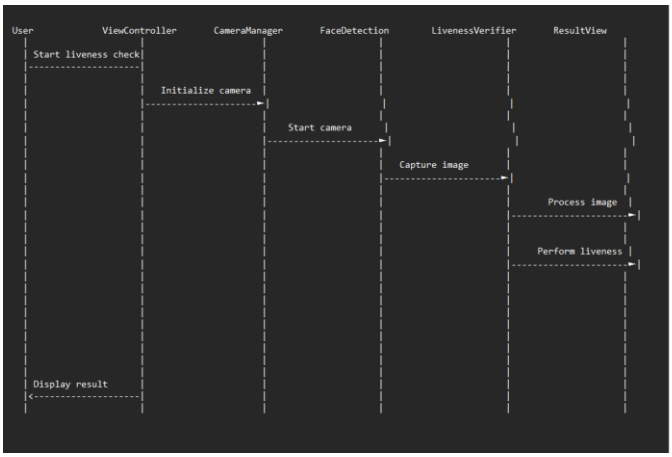

Figure 3.9: Liveness Checking Sequence Diagram

Chapter 4

Implementation

4.1 NID Text Extraction:

Step 1:

Firstly, we import some of our important frameworks. Then we have to check liveness with camera preview.

```
private var videoCapture: CameraPreview?
var capturedImage: UIImage?
var isSmartNID = true
```

Step 2:

Then we create a class with a viewcontroller for view in UI. Then we took two outlets in the main navigation controller.

```
@IBOutlet weak var cameraPreviewView: UIView!
@IBOutlet weak var captureButton: UIButton!
```

```
private func setUpCamera() {
    videoCapture = CameraPreview()
    videoCapture?.delegate = self
    videoCapture?.fps = 15
```

Step 3 :

To check internal properties we created a function. We checked which type of NID card we took.

```
func confirmationNIDType()  {
    var alertConfiramtion = UIAlertController(title: "NID Type", message: "Please
        Select your NID Type", preferredStyle: .alert)
    alertConfiramtion.addAction(UIAlertAction(title: "Smart Card", style: .default,
        handler: { alert in
        self.isSmartNID = true
    }))
    alertConfiramtion.addAction(UIAlertAction(title: "Old NID", style: .default,
        handler: { alert in
        self.isSmartNID = false
    }))

    self.present(alertConfiramtion, animated: true)
}
```

Step 4:

Then we set up the video and captured the NID card with an action button.

```
@IBAction func captureButtonAction(_ sender: UIButton) {
    if let capturedImage = capturedImage{
        //saveImageToDocumentDirectory(image: capturedImage, fileName: "NID")
```

Step 5:

It stored ID holder name, date of birth, and ID card number. In experimental results we saw the real output of two NID cards.

```
extension ViewController: CameraPreviewDelegate {

    func videoCapture(_ capture: CameraPreview, didCaptureVideoFrame pixelBuffer:
        CVPixelBuffer?, timestamp: CMTime) {
        if let pixelBuffer = pixelBuffer {
            self.convertPixelBufferToImage(pixelBuffer: pixelBuffer)
        }
    }

}
```

4.2 Liveness checking:

Step 1:

Firstly, we import some of our important frameworks, then we add some attribute strings.

```
import UIKit
import CoreMedia
import MLImage
import MLKit

enum LivenessSteps: String{
    case smile, leftTurned, rightTurned, eyeBlink
}

class ViewController: UIViewController {
```

Step 2:

Then we create some outlets for the camera, preview, and main navigation controller.

```
@IBOutlet weak var cameraPreviewView: UIView!
@IBOutlet weak var topLayerImageView: UIImageView!
@IBOutlet weak var captureButton: UIButton!
@IBOutlet weak var instructionLabel: UILabel!
```

Step 3:

Then we use some functions to create Internal properties,

```
private var videoCapture: CameraPreview?
var livenessSteps: LivenessSteps?

override func viewDidLoad() {
    super.viewDidLoad()

    setUpCamera()
    livenessSteps = .smile
    instructionLabel.text = "Smile Please 😊"
}
```

Step 4:

Then we set up our video,

```
private func setUpCamera() {
    videoCapture = CameraPreview()
    videoCapture?.delegate = self
    videoCapture?.fps = 15
```

Step 5:

And finally, we use the liveness check function to detect the faces,

```
LivenessCheck.shared.detectFaces(image: uiImage) { faces in
    if let faces = faces{
        guard let livenessSteps = self.livenessSteps else{return}

        switch livenessSteps {
        case .smile:
```

```
extension ViewController: CameraPreviewDelegate {

    func videoCapture(_ capture: CameraPreview, didCaptureVideoFrame pixelBuffer:
        CVPixelBuffer?, timestamp: CMTime) {
        if let pixelBuffer = pixelBuffer {
            self.convertPixelBufferToImage(pixelBuffer: pixelBuffer)
        }
```

Chapter 5

Result

5.1 NID Text Extraction

In experimental results, we saw the real output of NID cards.

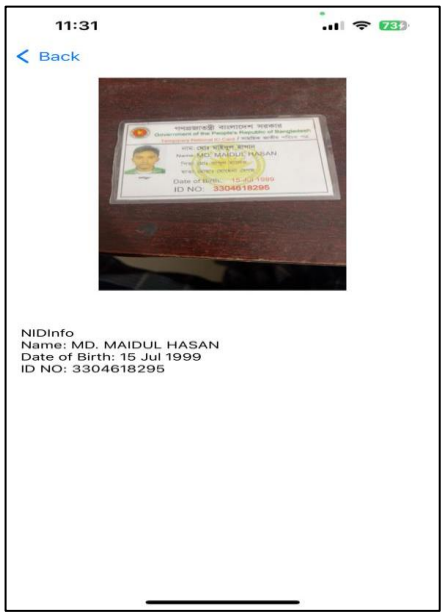

Figure 5.1: NID Text Extraction Output

5.2 Liveness Checking

we integrated four steps to complete the process, Every action will be instructed by the app at the top of the camera preview.

Step 1:

In the first step the person will smile, then the XCode output will show smile,

Figure 5.2: Smile Output

Step 2:

In the second step, the person will turn his head left then the XCode output will show left turned.

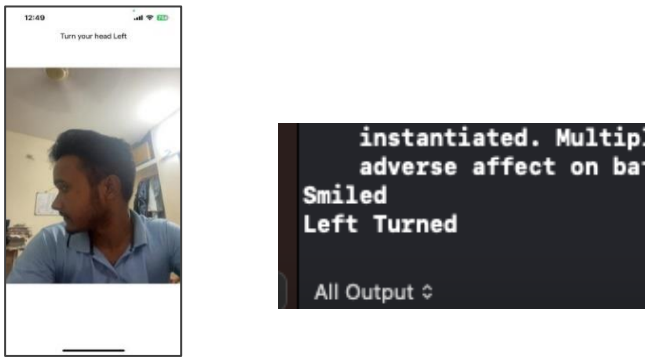

Figure 5.3: Left Turned Output

Step 3:

In the third step, the person will turn his head right, then the XCode output will show right turned.

Figure 5.4: Right Turned Output

Step 4:

In the fourth step, the person will blink his eyes then the XCode output will show eye blinked.

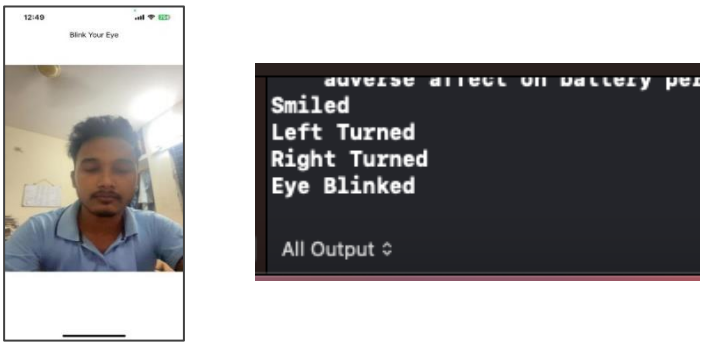

Figure 5.5: Eye Blinking Output

Chapter 6

Conclusion

6.1 Advantages

Our project, "Liveness Checking and NID Text Extraction," stands as a significant leap forward in the realm of identity verification and document processing, offering several key advantages:

i. **Enhanced Security**: Real-time liveness checks are incorporated into our app to increase identity verification and lower the danger of unauthorized access.

ii. **Efficiency Boost:** Data entry procedures are made more efficient by NID text extraction, which also reduces mistakes and speeds up verification.

iii. **User-Centric Design**: Our user-friendly software, which is optimized for iOS devices, puts convenience and accessibility first.

iv. **Comprehensive Solution**: A comprehensive solution to identity verification is provided by addressing authentication issues and document processing on one platform.

v. **Cross-Platform Potential**: The framework established for iOS programming paves the way for conceivable future extensions to other significant platforms.

vi. **Continuous Improvement:** Our dedication to incremental improvements based on user input keeps the app at the cutting edge of security and usability.

vii. **Adaptive Security:** The app adapts its authentication intensity based on risk, balancing security and convenience.

viii. **Future-Ready**: Future improvements and sophisticated features are made possible by using AI and machine learning.

ix. **Versatile Application:** The app's functionality may be expanded to include more sorts of identifying documents in addition to NID, increasing its usefulness.

Using iOS development, "Liveness Checking and NID Text Extraction" provides a revolutionary improvement in the disciplines of identity verification and document processing. It is a crucial tool for businesses and individuals looking for reliable and approachable solutions in an increasingly

digital environment because of its benefits, which not only strengthen the security of authentication procedures but also improve efficiency, usability, and compliance.

6.2 Future Enhancement

There is no limit to what we can achieve and develop new features on top of the existing project.

However, following enhancement proposals should be taken into consideration in the next phase of development:

6.2.1 NID Text Extraction:

Future features for an NID (National ID) text extraction app project using iOS development could enhance its functionality and utility. Here are some potential future features to consider:

i. **Multi-Language Support:** Add other languages and character sets to the app's text extraction capabilities so that it may be used in a variety of international scenarios.

ii. **Document Type Recognition:** Implement the capability to identify and extract text from a variety of identity papers, including NID documents as well as passports, licenses, and visas.

iii. **Data Validation:** Reduce mistakes in the verification process by integrating data validation tests to assure the correctness and legitimacy of extracted information.

iv. **Enhanced OCR Accuracy**: Increase the accuracy of the Optical Character Recognition (OCR) algorithms constantly, especially when dealing with handwritten or imperfect text.

v. **Document Image Enhancement:** To enhance OCR outcomes, create methods for boosting the quality of document pictures, such as cropping, altering brightness, or eliminating background noise.

vi. **Cloud Integration**: Permit users to safely store and retrieve extracted data on the cloud, making it usable on many platforms and devices.

vii. **Data Export:** Provide choices for transferring extracted data to other programs or systems in a variety of formats, such as PDF or CSV.

viii. **Offline Mode:** Create an offline data extraction option that enables users to extract text without an internet connection.

ix. **Integration with ID Verification Services:** Work together with identity verification services to make it simple for users to utilize the data that has been retrieved to validate their identities.

x. **Voice Assistant Integration**: Adding voice control to data extraction and verification would increase the app's usability for users of all abilities.

xi. **Machine Learning for Data Extraction:** Improve the app's capability to precisely extract data from various document formats and layouts by implementing machine learning methods.

These future features can help transform our NID text extraction app into a comprehensive and versatile tool that not only extracts text but also enhances user convenience, data security, and the overall user experience.

6.2.2 Liveness Checking:

Future features for a "Liveness Checking" app project using iOS development can enhance its capabilities in ensuring the real-time presence and authenticity of users. Here are some potential future features to consider:

i. **Multi-Modal Liveness Checks:** By combining several biometric modalities, such as voice patterns, facial expressions, and behavioural biometrics (e.g., gesture recognition), the range of liveness tests can be increased.

ii. **Continuous Authentication:** Implement continuous authentication to track and re-identify users during ongoing sessions in order to quickly identify illegal or questionable behaviour.

iii. **Liveness Challenges**: To avoid static picture or video attacks, include arbitrary liveness challenges throughout the authentication process, such as asking the user to take certain behaviours (such as blinking, smiling, or nodding)

iv. **Cross-Device Compatibility:** Allow users to easily execute liveness tests on a variety of iOS devices, including Macs, iPhones, and iPads.

v. **Liveness Logs and Auditing:** To improve security and compliance, keep thorough logs of liveness checks and authentication attempts for auditing and forensic use.

vi. **Integration with Identity Verification Services:** Offer a complete identity validation solution that includes liveness checks by working with identity verification services.

vii. **Real-time Reporting:** Transparency and security awareness are promoted by giving users real-time information and insights on their authentication and liveness check actions.

viii. **Privacy Controls:** To ensure user data safety and adherence to data privacy rules, privacy controls should be improved. These controls should include data encryption and user consent processes.

These future features can significantly enhance the effectiveness, security, and user experience of your "Liveness Checking" app, making it a valuable tool in the realm of identity verification and access control.

6.3 Limitations

While "Liveness Checking" apps developed using iOS development can provide enhanced security and user authentication, they also have certain limitations and challenges. Here are some common limitations of such apps:

i. **Hardware Dependency:** Liveness verification frequently depends on specialized technology, including cameras and sensors. The quality and capabilities of the hardware on the user's iOS device may have an impact on the app's usefulness.

ii. **False Positives and Negatives:** Liveness checks sometimes result in false positives (mistaking a real user for a fraudster) or false negatives (failing to catch attempted fraud). It might be difficult to maintain precision while reducing these mistakes.

iii. **User Experience:** The user experience may be significantly impacted by excessively invasive or frequent liveness checks. Repeated inspections could be tedious and time-consuming for users, which might make them frustrated.

iv. **Device Compatibility:** It's possible that not all iOS devices can use the same liveness checking capabilities or have the required sensors. Different devices may suffer varying user interfaces and security levels as a result.

v. **Privacy Concerns:** Liveness verification frequently entails the collection and processing of biometric information, which creates privacy issues. Users could be reluctant to allow authorization for access to their biometric data, and apps need to treat user data cautiously.

vi. **Spoofing Techniques:** To get beyond liveness checks, determined attackers may use sophisticated spoofing tactics, such as employing high-quality photos or videos. A ongoing difficulty is keeping up with the development of spoofing techniques.

vii. **Accessibility:** Users with disabilities or those who would struggle with particular forms of identification, including those who have problems with facial recognition, may face difficulties with liveness checks.

viii. **Maintenance and Updates:** It takes continual development and maintenance work to keep the program current with changing security risks, hardware developments, and user expectations.

ix. **Costs:** An efficient liveness monitoring software may be expensive to create and run, especially when considering the requirement for ongoing upgrades and enhancements.

x. **Scalability:** If the software is intended for general usage, it must be scalable enough to handle a big user base without sacrificing speed.

References

1. https://www.incognia.com/the-authentication-reference/what-is-liveness-detection#:~:text=Also%20known%20as%20%E2%80%9Canti%2Dspoofing,or%20recreated%20image%20of%20one.

2. https://www.researchgate.net/publication/258549622_Extraction_of_Words_from_the_National_ID_Cards_for_Automated_Recognition

3. https://developers.innovatrics.com/digital-onboarding/docs/functionalities/document/ocr/#:~:text=Optical%20Character%20Recognition%20(OCR)%20functionality,image%20data%20from%20those%20images.

4. https://www.innovatrics.com/glossary/liveness-detection/

5. https://www.netguru.com/blog/what-is-xcode-and-how-to-use-it

6. https://www.educative.io/blog/swift-programming

7. https://blogs.infosys.com/digital-experience/emerging-technologies/google-ml-kit-for-on-device-machine-learning.html#:~:text=ML%20Kit%20is%20free%20of,Lite%2C%20and%20Neural%20Network%20APIs.

8. https://medium.com/@tristate/why-to-choose-ios-platform-for-app-development-15fb43575f13

9. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

10. https://www.techtarget.com/searchdatamanagement/definition/entity-relationship-diagram-ERD#:~:text=An%20entity%20relationship%20diagram%20(ERD,information%20technology%20(IT)%20system.

11. https://creately.com/diagram/example/gwp1smhc1/ios-uml#:~:text=The%20iOS%20UML%20diagram%20template,Unified%20Modeling%20Language%20(UML).

12. https://www.ibm.com/docs/en/rsm/7.5.0?topic=uml-sequence-diagrams

13. https://www.researchgate.net/publication/262992608_An_Overview_of_Face_Liveness_Detection

14. https://www.researchgate.net/publication/337664348_Insight_on_face_liveness_detection_A_systematic_literature_review

15. https://www.fraud.com/post/liveness-detection

16. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-optical-character-reader-ocr/

17. https://www.tutorialspoint.com/advantages-and-disadvantages-of-optical-character-reader-ocr

18. https://www.klearstack.com/id-card-ocr/

19. https://www.tutorialspoint.com/advantages-and-disadvantages-of-optical-character-reader-ocr#:~:text=OCR%20technology%20is%20limited%20to,and%20other%20types%20of%20data.

20. https://waqasmushtaq.medium.com/is-ocr-still-a-challenge-reasons-and-way-forward-in-2023-7fa66670ad83#:~:text=One%20of%20the%20main%20challenges,for%20OCR%20systems%20to%20interpret

21. https://www.linkedin.com/advice/0/what-current-challenges-limitations-face#:~:text=Variability%20of%20faces,with%20different%20scenarios%20and%20conditions.