Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

SKILL BASED EVALUATION REPORT

Academic Year 2025-2026

ODD SEMESTER

Name : **DAVID PAUL P**

Register Number : **URK23CS8005**

Course Code : **24CS2001**

Course Name : **AI IN TELEMEDICINE AND HEALTHCARE**

NOV/DEC 2025

INTRODUCTION ABOUT THE SUBJECT

Artificial Intelligence (AI) has emerged as a transformative technology across various domains, and healthcare is one of the most promising fields benefiting from its advancements. AI in telemedicine and healthcare enables faster diagnostics, personalized treatment plans, remote monitoring, and improved patient outcomes. With the growing demand for accessible and efficient healthcare services, AI-driven solutions are helping clinicians make more accurate decisions while reducing human error and operational costs.

The course AI in Telemedicine and Healthcare (24CS2001) explores the application of AI technologies such as machine learning, computer vision, natural language processing, and predictive analytics in medical contexts. It focuses on integrating AI with telemedicine platforms to enhance patient care, monitor health conditions remotely, and facilitate early disease detection.

In this project, “AI for Hair Regrowth Assessment,” AI techniques are applied to evaluate hair growth patterns, monitor treatment effectiveness, and provide personalized recommendations. By analyzing images and data over time, the system can accurately assess hair regrowth progress, offering a convenient and non-invasive tool for both patients and clinicians. This application demonstrates how AI can bridge technology and healthcare, improving patient engagement and clinical decision-making in dermatology.

INTRODUCTION ABOUT THE COURSE

The course AI in Telemedicine and Healthcare (24CS2001) focuses on the integration of Artificial Intelligence (AI) technologies into the healthcare and telemedicine sector to enhance patient care, clinical decision-making, and operational efficiency. It begins with the evolution of telemedicine, tracing its development from basic remote consultations to modern platforms that leverage AI for diagnostics, monitoring, and patient management.

The course explores the role of AI in healthcare, emphasizing applications such as predictive analytics, medical image processing, disease detection, and personalized treatment recommendations. Students study various AI and machine learning models, including deep learning, neural networks, and natural language processing, to understand how these algorithms can extract insights from complex medical data.

Additionally, the course covers sensors and IoT devices that collect continuous patient data, enabling real-time monitoring and integration with AI systems for automated analysis. It also addresses the challenges of AI in healthcare, including data privacy, ethical considerations, reliability, and system integration, which are critical for safe and effective implementation.

Practical learning through projects allows students to apply AI models for real-world healthcare solutions. This includes analyzing patient data, monitoring treatment progress, and providing actionable insights, highlighting how AI can improve accessibility, efficiency, and outcomes in modern healthcare.

ONLINE COURSE CERTIFICATION

HAIR REGROWTH ASSESSMENT

A PROJECT REPORT

Submitted by

DAVID PAUL P

URK23CS8005

DIVISION OF COMPUTER SCIENCE AND ENGINEERING

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed-to-be-under Sec-3 of the UGC Act, 1956)
Karunya Nagar, Coimbatore - 641 114. INDIA

NOV 2025

Abstract

The project implements an AI-powered Hairline Tracker designed to assess and monitor hair regrowth. Users can either capture a live image using a webcam or upload existing photos. The AI model detects the hairline, calculates critical metrics such as forehead ratio, hairline type, hair density, symmetry, and distances, and stores these for analysis. The system provides a visualization of hair regrowth progress over time, enabling users and clinicians to evaluate treatment effectiveness. The project also includes batch processing, historical data management, and progress tracking reports, demonstrating a practical application of AI in telemedicine.

Introduction

Artificial Intelligence is increasingly shaping modern telemedicine and healthcare systems, offering faster, more accurate, and non-invasive diagnostics. Hair loss affects millions worldwide, and effective monitoring of hair regrowth is essential for personalized treatment. Traditional assessment methods are subjective and prone to human error, whereas AI-based systems standardize evaluation using measurable metrics.

Objectives of the Project:

- Automated detection and assessment of hairline.
- Measurement of forehead ratios, hairline density, symmetry, and height.
- Visualization of hair regrowth over time.
- User-friendly interface for both webcam capture and image uploads.
- Storage and tracking of historical user data for longitudinal analysis.

System Architecture & Methodology

1.1 Architecture Overview

1. User Interface: Webcam capture or photo upload for user input.
2. Hairline Detection Module (HairlineDetector): Uses deep learning to identify hairline and calculate metrics such as height, density, and symmetry.
3. Progress Tracking Module (ProgressTracker): Stores analysis results and generates progress visualizations.

4. Data Management Module (DataManager): Handles directory structures, saves input/output images, validates image quality, and manages historical data.

1.2 AI & ML Models Used

- Deep Learning Models: For facial and hairline landmark detection. Likely trained on datasets like CelebA and WIDER FACE.
- Image Processing: To calculate forehead ratio, hairline distance, hair density, and symmetry.
- Visualization: Overlays results on the original image and plots growth over time for clear user feedback.

1.3 Workflow

1. User opens the software and selects webcam capture or image upload.
2. HairlineDetector analyzes the image:
    - Detects the face and hairline.
    - Calculates metrics: type, height, density, symmetry, distances.
3. Results are saved using DataManager.
4. ProgressTracker generates visualizations for long-term monitoring.
5. Users can view history, generate reports, or process batch images.

Implementation Details

5.1 Tools and Technologies

- Programming Language: Python 3.x
- Libraries & Frameworks: OpenCV (image capture and processing), TensorFlow/PyTorch (deep learning), Matplotlib/Seaborn (visualizations), NumPy (data handling)

5.2 Features Implemented

1. Webcam Image Capture: Provides real-time video feed for users to capture photos.
2. Single Image Processing: Upload and analyze images for hairline assessment.

3. Batch Image Processing: Process multiple images from a folder automatically.
4. Hairline Analysis: Calculates hairline type, forehead ratio, symmetry, density, and other facial metrics.
5. Progress Visualization: Graphs showing improvement over time.
6. User History Management: Maintains analysis records for each user with timestamps.
7. Dataset Management: Supports downloading sample datasets or creating sample images for testing AI models.

5.3 Code Structure

- main.py: The main executable script with menu-driven options.
- hairline_detector.py: Contains the AI model and functions to detect and measure hairline metrics.
- progress_tracker.py: Handles tracking, report generation, and visualization.
- data_manager.py: Manages storage, validation, and retrieval of images and analysis results.

```
import cv2
import os
import sys
from datetime import datetime
from hairline_detector import HairlineDetector
from progress_tracker import ProgressTracker
from data.data_manager import DataManager

class HairlineTrackerApp:
    def __init__(self):
        self.detector = HairlineDetector()
        self.tracker = ProgressTracker()
        self.data_manager = DataManager()
        print("🚀 Hairline Tracker initialized successfully!")

    def setup_environment(self):
        """Setup the complete environment"""
        print("🔧 Setting up environment...")
        self.data_manager.setup_directories()
        self.data_manager.create_sample_images()
        print("☑ Environment setup complete!")

    def take_photo(self):
```

```
        """Take photo with webcam and analyze immediately"""
        cap = cv2.VideoCapture(0)

        if not cap.isOpened():
            print("❌ Cannot access webcam. Make sure it's connected and not used by another application.")
            return

        print("📷 Webcam activated!")
        print("💡 Instructions:")
        print("   - Look straight at the camera")
        print("   - Make sure your face is well-lit")
        print("   - Press SPACEBAR to capture photo")
        print("   - Press ESC to cancel")

        while True:
            ret, frame = cap.read()
            if not ret:
                print("❌ Failed to capture image from webcam")
                break

            # Display the live video
            cv2.imshow('Press SPACEBAR to capture - ESC to cancel', frame)

            key = cv2.waitKey(1) & 0xFF
            if key == ord(' '):  # SPACEBAR to capture
                user_id = input("Enter user ID for this photo: ").strip() or "webcam_user"

                # Save the captured image
                timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
                save_path = f"data/input/raw_images/{user_id}_webcam_{timestamp}.jpg"
                success = cv2.imwrite(save_path, frame)

                if success:
                    print(f"☑ Photo saved: {save_path}")

                    # Analyze the photo immediately
                    print("🔄 Analyzing hairline...")
                    result = self.detector.analyze_hairline(frame)

                    if result:
                        print("🎉 Analysis successful!")
```

```
                        print(f"   Hairline Type: {result['hairline_type']}")
                        print(f"   Height: {result['hairline_height']:.3f}")
                        print(f"   Density: {result['density_score']:.3f}")

                        # Save results
                        self.tracker.save_analysis(user_id, timestamp, result)
                        self.data_manager.save_analysis_result(result, user_id,
timestamp)

                        # Show and save visualization
                        vis_image = self.detector.visualize_analysis(frame,
result)
                        self.data_manager.save_visualization(vis_image, user_id)
                        cv2.imshow('Hairline Analysis Results', vis_image)
                        print("📊 Press any key to close results...")
                        cv2.waitKey(0)

                        print("💾 All results saved to output folders!")
                    else:
                        print("❌ No face detected in the photo.")
                        print("   Please ensure:")
                        print("   - Your face is clearly visible")
                        print("   - Good lighting conditions")
                        print("   - Front-facing pose")
                else:
                    print("❌ Failed to save photo")

                break
            elif key == 27:  # ESC to cancel
                print("❌ Capture cancelled")
                break

        cap.release()
        cv2.destroyAllWindows()

    def process_single_image(self, image_path=None, user_id=None):
        """Process a single image and analyze hairline"""
        if image_path is None:
            image_path = input("Enter image path: ").strip()

        if user_id is None:
            user_id = input("Enter user ID (default: 'user1'): ").strip() or
"user1"

```

```
            print(f"📸 Processing image: {image_path}")

            # Validate image
            is_valid, message = self.data_manager.validate_image(image_path)
            if not is_valid:
                print(f"❌ Image validation failed: {message}")
                return None

            # Read image
            image = cv2.imread(image_path)
            if image is None:
                print(f"❌ Could not load image: {image_path}")
                return None

            # Save input image
            saved_path = self.data_manager.save_input_image(image, user_id)

            # Detect hairline and get metrics
            print("🔍 Analyzing hairline...")
            result = self.detector.analyze_hairline(image)

            if result:
                # Save results
                timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
                self.tracker.save_analysis(user_id, timestamp, result)

                # Save analysis result
                analysis_path = self.data_manager.save_analysis_result(result,
    user_id, timestamp)

                # Create and save visualization
                vis_image = self.detector.visualize_analysis(image, result)
                vis_path = self.data_manager.save_visualization(vis_image, user_id)

                print("☑️ Analysis completed successfully!")
                print(f"📊 Hairline Type: {result['hairline_type']}")
                print(f"📏 Height: {result['hairline_height']:.3f}")
                print(f"📈 Density: {result['density_score']:.3f}")
                print(f"⚖️  Symmetry: {result['symmetry_score']:.3f}")

                # Display results
                cv2.imshow('Hairline Analysis Results', vis_image)
                print("🖼️ Press any key to close the visualization window...")
                cv2.waitKey(0)
```

```
            cv2.destroyAllWindows()

            return result
        else:
            print("❌ Hairline analysis failed - no face detected")
            return None

    def process_batch_images(self, input_folder=None, user_id=None):
        """Process all images in a folder"""
        if input_folder is None:
            input_folder = input("Enter folder path (default:
'data/input/raw_images'): ").strip() or "data/input/raw_images"

        if user_id is None:
            user_id = input("Enter user ID (default: 'batch_user'): ").strip() or
"batch_user"

        print(f"🔄 Processing batch images from: {input_folder}")

        valid_images, invalid_images =
self.data_manager.batch_process_images(input_folder, user_id)

        if not valid_images:
            print("❌ No valid images found to process")
            return []

        results = []
        for image_path in valid_images:
            print(f"🔍 Processing: {os.path.basename(image_path)}")
            result = self.process_single_image(image_path, user_id)
            if result:
                results.append(result)

        print(f"\n📊 Batch processing complete!")
        print(f"☑️ Successful analyses: {len(results)}")
        print(f"❌ Failed analyses: {len(valid_images) - len(results)}")

        return results

    def track_progress(self, user_id=None):
        """Generate progress report for a user"""
        if user_id is None:
            user_id = input("Enter user ID (default: 'user1'): ").strip() or
"user1"
```

```
        print(f"📈 Generating progress report for: {user_id}")

        report = self.tracker.generate_report(user_id)

        if report:
            # Save progress report
            report_path = self.data_manager.save_progress_report(report, user_id)
            print(f"☑ Progress report generated: {report_path}")

            # Export user data
            export_path = self.data_manager.export_user_data(user_id)
            print(f"📤 User data exported: {export_path}")

            return report
        else:
            print("❌ No data available for progress tracking")
            return None

    def show_user_history(self, user_id=None):
        """Show user's analysis history"""
        if user_id is None:
            user_id = input("Enter user ID (default: 'user1'): ").strip() or
"user1"

        print(f"📋 Analysis history for: {user_id}")

        history = self.data_manager.get_user_history(user_id)

        if not history:
            print("   No analysis history found")
            return

        print(f"   Found {len(history)} analyses:")
        for i, analysis in enumerate(history, 1):
            print(f"   {i}. {analysis['timestamp']} - {analysis['filename']}")

    def download_sample_datasets(self):
        """Download sample datasets for training"""
        print("📥 Available Datasets:")
        print("1. CelebA - Large-scale Face Attributes")
        print("2. WIDER FACE - Face Detection Benchmark")
        print("3. Create Sample Images (Recommended for testing)")
```

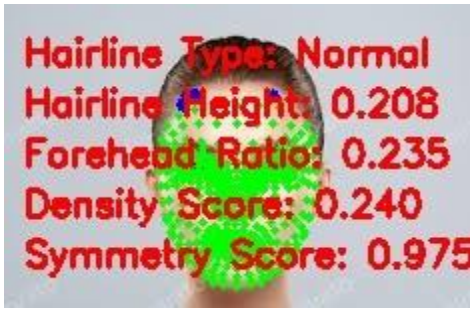
```
            dataset_choice = input("Choose dataset (1-3): ").strip()
            if dataset_choice == '1':
                print("📥 CelebA dataset: Visit http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html")
                print("   Download 'Align&Cropped Images' and extract to data/input/datasets/celeba/")
            elif dataset_choice == '2':
                print("📥 WIDER FACE dataset: Visit http://shuoyang1213.me/WIDERFACE/")
                print("   Download WIDER_train.zip and extract to data/input/datasets/wider_face/")
            else:
                self.data_manager.create_sample_images()

    def main():
        """Main function with menu interface"""
        app = HairlineTrackerApp()

        while True:
            print("\n" + "="*50)
            print("🧬 HAIRLINE TRACKER - AI TELEMEDICINE")
            print("="*50)
            print("1. Setup Environment (First Time)")
            print("2. Take Photo with Webcam")
            print("3. Process Single Image")
            print("4. Process Batch Images")
            print("5. Track Progress")
            print("6. Show History")
            print("7. Download Sample Datasets")
            print("8. Exit")
            print("-"*50)

            choice = input("Enter your choice (1-8): ").strip()

            if choice == '1':
                app.setup_environment()

            elif choice == '2':
                app.take_photo()

            elif choice == '3':
                app.process_single_image()

            elif choice == '4':
```

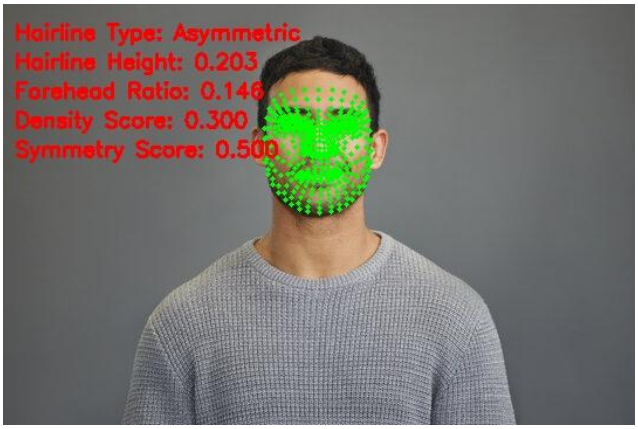
```
            app.process_batch_images()

        elif choice == '5':
            report = app.track_progress()
            if report:
                print("\n" + "="*50)
                print(report)
                print("="*50)

        elif choice == '6':
            app.show_user_history()

        elif choice == '7':
            app.download_sample_datasets()

        elif choice == '8':
            print("👋 Thank you for using Hairline Tracker!")
            break

        else:
            print("❌ Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Screenshots of Outcomes

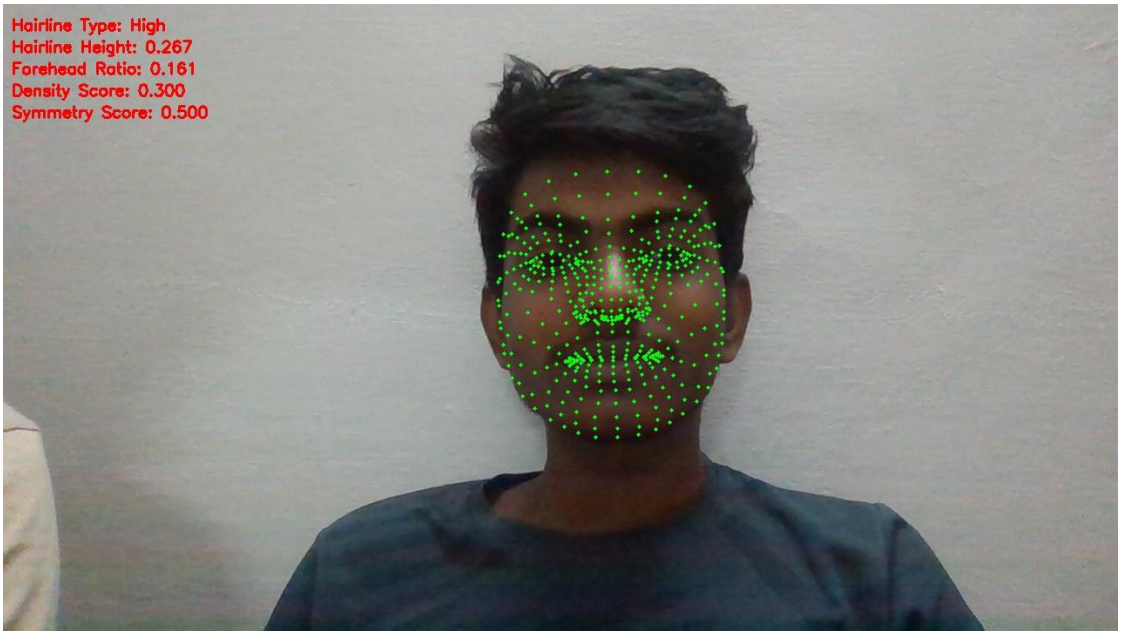
1. Webcam Interface: Shows live feed with instructions to capture the photo.
2. Hairline Analysis Result: Annotated image with metrics displayed.
3. Progress Tracking Visualization: Graphs showing hair regrowth over time for a specific user.
4. Batch Processing Summary: Table of processed images, successes, and failures.
5. User History: Historical data of all analyses with timestamps and image references.

1. 

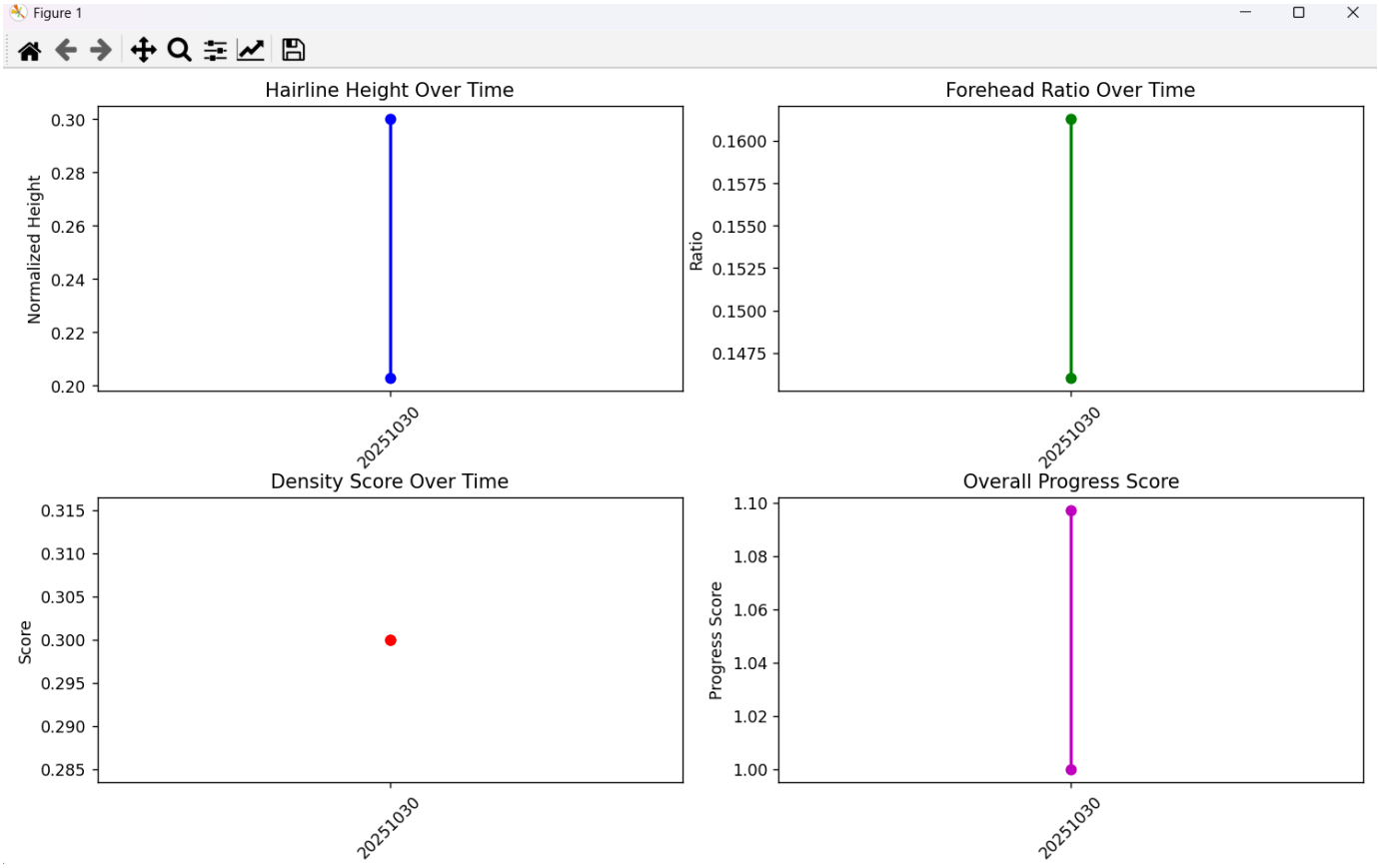
2. 

3. 

4. Camera based:



5.Report



Results and Discussion

- Accuracy: AI accurately detects hairline and calculates key metrics.
- Visualization: Clear progress tracking allows easy understanding of treatment effectiveness.
- Batch Processing: Enables efficient handling of multiple images for clinics or research purposes.
- Limitations: Dependent on proper lighting and frontal face orientation; different hair types may need further training.
- Applications: Telemedicine, dermatology clinics, patient self-monitoring, and research on hair regrowth treatments.

Conclusion

- The Hairline Tracker successfully integrates AI into telemedicine for hair regrowth assessment.
- Provides a non-invasive, automated, and standardized solution.
- Facilitates real-time monitoring, progress visualization, and historical data tracking.
- Future scope includes mobile app integration, predictive analytics, and support for a wider variety of hair types and lighting conditions.

References

- Research papers on AI in dermatology, facial landmark detection, and telemedicine applications.
- Open-source datasets: CelebA, WIDER FACE.
- Online documentation for OpenCV, TensorFlow, and PyTorch.
- AI and deep learning textbooks and online tutorials for model training and image processing.

EVALUATION SHEET

Reg.No : URK23CS8005

Name : DAVID PAUL P

Course code : 24CS2001

Course Name : AI IN TELEMEDICINE AND HEALTHCARE

S.No	Rubrics	Maximum Marks	Marks Obtained
1	Online Course Completion	10	
2	Project - Deployment of a model	25	
3	Project Presentation	5	
Total		40	

Signature of the Faculty-in-charge