SRM UNIVERSITY, ANDHRA PRADESH

SUMMER INTERNSHIP COURSE, June 2025

**WEEKLY DIARY REPORT**

**(B.Tech. STUDENT BATCH 2023-27)**

**(To be submitted to Faculty Mentor over mail with CC to Industry mentor)**

Name of Student: BHAVYA SRI PULIKHANDAM

ID NO: AP23110010929

Company Name (For industry internship):  TIMING TECHNOLOGIES

Name of Faculty Mentor:Dr.Abijith Dasagupta

|  |  |  |
| --- | --- | --- |
| Week 1 From 20/05/2025 to 23/05/2025 | | |
| S. No | Field | Answer |
| 1 | Project Title | Automated Participant Identification Using Face Recognition, Bib Detection |
| 2 | Project Description | The project focuses on developing an automated system for participant identification in sports and athletic events by combining advanced face recognition and bib number detection techniques. The system leverages computer vision and image processing to capture real-time images of participants, accurately detect their faces, and extract bib numbers using Optical Character Recognition (OCR).  To ensure high accuracy and efficiency, the system is integrated with RFID technology, which provides an additional layer of identity validation. This multi-modal approach reduces manual effort, enhances event management, and minimizes identification errors.  The solution enables real-time participant verification, which is essential for applications such as race timing, attendance tracking, and result management. By automating the detection process, the system significantly improves operational speed, accuracy, and overall event experience. |
| 3 | Outline of the Solution | Sure! Here's a **short and crisp Outline of the Solution** based on your project:  **Outline of the Solution (Short Version)**  The system detects and verifies participants involved in "wrong tag" events by combining face recognition and bib number identification:   1. **Face Encoding Database**    * Loads participant registration images from .pickle files.    * Extracts and stores face encodings along with bib numbers and roll numbers. 2. **Wrong Tag Image Processing**    * Retrieves unprocessed wrong tag frames from MongoDB and GridFS.    * Detects faces in these frames and compares them with known encodings. 3. **Participant Verification**    * If a matching face is found, retrieves bib and roll number.    * Stores verified results in the bib\_detection\_results collection.    * Marks frames as processed to avoid duplicates. 4. **Logging and Reporting**    * Logs system performance metrics.    * Generates an Excel report summarizing detection results. |
| 4 | Design of the Solution | 1.Loads participant face encodings and bib numbers from .pickle files.  2.Retrieves unprocessed wrong tag images from MongoDB and GridFS.  3.Detects faces in images and compares with known encodings.  4.If a match is found, stores bib number, roll number, and confidence in the database.  5.Marks processed frames to avoid duplicates.  6.Generates an Excel report with detection results and performance metrics. |
| 5 | Hardware and Software Requirements to execute the project | **Hardware Requirements**  1.Laptop/Desktop with minimum **8 GB RAM**, **i5/Ryzen 5 processor**, 500 GB storage  2.Optional: Webcam or RFID Reader for real-time input  **Software Requirements**  1**.**Python 3.8+, OpenCV, face\_recognition, numpy, pandas  2.MongoDB with GridFS  3.pymongo, pickle, openpyxl for data handling and reporting  4.Windows, Linux, or macOS |
| 6 | Environment setup windows/linux/Raspberry pi/Arduino | Windows 10/11 or Linux  **Software Requirements:**  Python 3.8 or higher  MongoDB with GridFS for image storage  **Python Libraries to Install:**  pip install opencv-python face\_recognition numpy pandas pymongo gridfs openpyxl |
| 7 | Concepts Used (Functions, header files, data types, and concepts (Loops, arrays, conditional statements, etc.). Explanation of the concepts). For Hardware projects also explain with respect to the code being developed | **1.Functions** Used to divide the project into reusable blocks like loading data, comparing faces, and saving results Example: compare\_faces(), load\_pickle(), process\_wrong\_tag\_frames()  **2.Header/FileLibraries** cv2 – For image processing face\_recognition – For face detection and comparison numpy – For handling image arrays pymongo and gridfs – For database operations pandas – For creating Excel reports pickle – For loading saved participant data  **3.DataTypesUsed** Strings, integers, lists, dictionaries, numpy arrays  **4.Programming Concepts** Loops: Used to process multiple images Conditional Statements: Used to check matching faces and handle errors Error Handling: Used to manage processing failures safely File Handling: Reading data files and saving reports  **5.HardwareRelevance** The project mainly uses stored image frames but can connect to cameras and RFID readers in real-time setups |
| 8 | Testing & Validation (Boundary tests and boundaries of inputs. Possible inputs and corresponding outputs). | | **Input Type** | **System Output** | | --- | --- | | Registered face image | Bib number and roll number detected, record stored | | Unregistered face image | No match found, system skips to next frame | | No face in image | Frame skipped, marked as processed | | Blurred/partial face image | Low confidence result or skipped frame | | Missing or corrupted image | Error logged, system continues | |
| 9 | Testing Material (Screenshots of working outputs. Images in case of hardware project |  |
| 10 | User Manual | **Project Title:**  Automated Facial Verification and Bib Number Detection System for Wrong Tag Identification  **Steps to Use the System**  **1. Prerequisites**   * Install Python 3.8 or higher * Install required libraries using: * pip install opencv-python face\_recognition numpy pandas pymongo gridfs openpyxl * Setup MongoDB and GridFS for image storage * Place .pickle files containing participant registration data in the specified folder   **2. Configuration Setup**   * Edit mongodb\_config.properties file with correct database details:   + mongo\_uri   + database\_name   + petdate   + encoding\_threshold (optional, default is 0.6)   **3. Running the System**   * Open terminal or command prompt * Run the main Python script: * python your\_script\_name.py * The system will automatically:   + Load participant face encodings from .pickle files   + Fetch unprocessed wrong tag frames from the database   + Perform face detection and comparison   + Store matched results into the database   + Mark frames as processed   **4. Output Generated**   * Verified bib numbers and roll numbers stored in MongoDB * Processing logs saved to console for real-time tracking * Excel report generated as facial\_logs.xlsx with detection details   **5. System Shutdown**   * Once processing is complete, the system automatically closes the database connection   **Important Notes**   * Ensure .pickle files contain valid images with clear participant faces * The system handles errors and skips invalid or unreadable images * Camera and RFID readers can be integrated externally if real-time input is required |
| 11 | Technical Documentation | **Project Title:**  Automated Facial Verification and Bib Number Detection System for Wrong Tag Identification  **System Overview**  The project detects participants involved in wrong tag events by comparing their face images with pre-registered data using face recognition techniques. Bib numbers and roll numbers are automatically verified and results are stored in MongoDB.  **Main Components**  **1. Face Encoding Database**   * Loads .pickle files containing face images and participant details * Generates face encodings for future comparison   **2. Image Processing Module**   * Retrieves unprocessed wrong tag frames from MongoDB and GridFS * Uses OpenCV and face\_recognition to detect faces   **3. Face Comparison and Validation**   * Matches detected faces with known encodings * If a match is found, retrieves bib number and roll number * Logs results in bib\_detection\_results collection   **4. Reporting Module**   * Generates Excel reports summarizing detection results and processing time   **Technologies Used**   * Python, OpenCV, face\_recognition * MongoDB with GridFS * Pandas for report generation * Pickle for data storage   **Error Handling**   * System safely skips invalid images or missing data * Logs errors without interrupting the process   **Performance Considerations**   * Confidence threshold controls match accuracy (default: 0.6) * System processes stored image frames; real-time camera input is optional |
| 12 | References |  |
| 13 | Daily Work Breakdown |  |
|  | Day 1 | Introduction to company, project overview, and meeting the team. |
|  | Day 2 | Basic orientation on the technologies used by the company |
|  | Day 3 | System setup: Installed required software (Python, MongoDB, VS Code) |
|  | Day 4 | Explored project files and folders, understood project workflow |
|  | Day 5 | Introduction to image processing basics, tried sample image loading |
| Week 2 From 02/06/25 to 06/06/25 | | |
|  | Day 1 | **Day 1: Project Setup & Environment Preparation** 1.Install required libraries (opencv-python, sqlite3, os, etc.) 2.Understand project flow — webcam access, image capture, database storage 3.Create folders for saving frames (captured\_frames) |
|  | Day 2 | 1.Created an SQLite database for storing frame metadata 2. Wrote and test SQL queries to create frame\_info table with fields:   * id (Primary Key) * timestamp (TEXT) * filename (TEXT) * Verify database file creation and schema |
|  | Day 3 | 1. Write code to access the webcam using cv2.VideoCapture() 2.Handle failure scenarios if webcam access fails |
|  | Day 4 | 1.Test reading frames from the webcam |
|  | Day 5 | 1.Save captured frames as image files with timestamped filenames |
| Week 3 From 09/06/25 to 13/06/25 | | |
|  | Day 1 | 1.Implement loop to capture multiple frames (total\_frames = 5)  2.Add cv2.imshow() to display frames live |
|  | Day 2 | 1.Implement delay using cv2.waitKey()  2.Allow early termination by pressing 'q' |
|  | Day 3 | 1.Run the full script end-to-end 2.Verify images saved in captured\_frames folder 3.Check frame\_metadata\_db for correct entries |
|  | Day 4 | 1.Handle edge cases (webcam not detected, folder permissions, etc)  2.Comment code for readability 3.Prepare a short explanation of code flow |
|  | Day 5 | 1.Demonstrate working project to supervisor or team  2.Clean up unnecessary files |
| Week 4 From 16/06/25 to 20/06/25 | | |
|  | Day 1 | 1.Installed required Python libraries: OpenCV, NumPy, face\_recognition, MongoDB client, GridFS  2.Understood project requirements: live face detection and secure storage system |
|  | Day 2 | 1.Configured MongoDB Database and GridFS for image storage  2.Tested MongoDB connections and basic document insertio |
|  | Day 3 | 1.Loaded registration data (images, bib numbers, roll numbers) from pickle files  2.Extracted facial encodings using face\_recognition library |
|  | Day 4 | 1.Loaded registration data (images, bib numbers, roll numbers) from pickle files  2.Extracted facial encodings using face\_recognition library |
|  | Day 5 | 1.Developed the live stream connection logic using OpenCV with retry mechanisms  2.Tested reading frames from RTSP/IP cameras and handled invalid streams |
| Week 5 From 23/06/25 to 27/06/25 | | |
|  | Day 1 | 1.Implemented face detection and comparison logic  2.Validated system with known encodings and tested matching functionality |
|  | Day 2 | 1.Integrated periodic frame capture (every 5th frame) from the live stream  2.Ensured system stability with reconnection logic for stream failures |
|  | Day 3 | 1.Successfully stored matched face images in MongoDB using GridFS  2.Saved relevant metadata (bib number, roll number, confidence, timestamps) |
|  | Day 4 | 1.Handled exceptions for frame issues, detection failures, and database errors |
|  | Day 5 | 1.figuring all the errors to even optimize code as during running cpu taking 100 percent so reserched on gpu to include in code to make code runs fast |
| Week 6 From 30/06/25 to 04/07/25 | | |
|  | Day 1 | 1.Conducted full system testing with live camera feed and sample pickle file  2.Verified that matched images and details were correctly stored in MongoDB |
|  | Day 2 | 1.working to update graphs in front end of bsf dashboard |
|  | Day 3 | 1.solved some errors while working on that used html,css |
|  | Day 4 | 1.updates those graphs using chart.js in that portal |
|  | Day 5 | 1.done research on multi threading and required hardwear |
| Week 7 From 07/07/25 to 11/07/25 | | |
|  | Day 1 | 1.created environment to run testing in my laptop |
|  | Day 2 | 1.added stated to code to find no of frames per second |
|  | Day 3 | 1.tested codes and drawn conclusions on the bases of outputs, efficiency |
|  | Day 4 | 1.tested code without multithreading and drawn conclusions based on that |
|  | Day 5 | 1.Drawn some analysis based on no of faces detecting and time analysis blw two codes |
| Week 8 From \_\_\_\_\_\_\_\_\_\_ to \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
|  | Day 1 |  |
|  | Day 2 |  |
|  | Day 3 |  |
|  | Day 4 |  |
|  | Day 5 |  |