**Real-time Stabilization and 3D Reconstruction of Hand Gestures and Finger Movement Traces Using LED-Equipped Gloves**



***SOFTWARE PROJECT MANAGEMENT PLAN (SPMP)***

**Version No. 1.0**

**Project Document Revision History**

| **VersionNumber** | **Date** | **Revision Author** | **Description of Revision** |
| --- | --- | --- | --- |
| 1.0 | 10/23/2024 | Team 5 | Original Published Version |
| 2.0 | 12/10/2024 | Team 5 | Updated Published Version |

| **Table of Contents** |
| --- |

[**1.0 Introduction 3**](#_heading=h.tykzobims4b1)

[1.1 Problem statement 4](#_heading=h.7h1ncbwgbxv2)

[1.2 Project scope 4](#_heading=h.2m9jeakjbgx4)

[1.2.1 Inclusions 4](#_heading=h.1bdjzueexbqy)

[1.2.2 Exclusions 4](#_heading=h.ql6geynr38sc)

[1.3 Major software functions 5](#_heading=h.pvxv2fe9gwd3)

[1.4 Performance/Behavior constraints 5](#_heading=h.tqf2cn6fxy23)

[1.5 Management and technical constraints 5](#_heading=h.zby2sxr8wotz)

[1.5.1 Management constraints 5](#_heading=h.2hzbllhu0n6t)

[1.5.2 Technical constraints 6](#_heading=h.ndfyth4bfh77)

[**2.0 Project Estimates 6**](#_heading=h.3c8vpw300wz)

[2.1 Historical data used for estimates 6](#_heading=h.ct8vdydi632p)

[2.2 Estimation techniques applied and results 6](#_heading=h.d6sjmcxdo7bw)

[2.2.1 Line-of Code Estimate 6](#_heading=h.zi0a3clhpr3q)

[2.2.2 Function Point Estimate 7](#_heading=h.w3mpuyjrz1ct)

[2.2.3 Tasks Estimate 8](#_heading=h.ya9vh1tcmatl)

[2.3 Estimation techniques applied and results 8](#_heading=h.qgcoyth7ix3)

[2.4 Project Resources 9](#_heading=h.whi3pfvhpmy5)

[**3.0 Risk Management 9**](#_heading=h.4f4m9mq2lbjl)

[3.1 Project Risk Table 9](#_heading=h.kqoz54qrjlgx)

[3.2 Overview of Risk Mitigation, Monitoring, Management 11](#_heading=h.lbdyvz6jq81i)

[**4.0 Project Schedule 11**](#_heading=h.39cxbmq20dhs)

[4.1 Project task set 11](#_heading=h.5i3zdevlcpt1)

[4.2 Task network 12](#_heading=h.ng2sgbykid1a)

[4.3 Timeline chart 14](#_heading=h.5rjrlyclcgyx)

[**5.0 Staff Organization 15**](#_heading=h.n0a3b49mwni3)

[5.1 Team structure 15](#_heading=h.iudz6vmfkiut)

[5.2 Management reporting and communication 15](#_heading=h.p0g2hmjmnbgs)

[**6.0 Tracking and Control Mechanisms 16**](#_heading=h.993fuqaawv0k)

[6.1 Quality assurance and control 16](#_heading=h.xdx70unvza3x)

[6.2 Change management and control 16](#_heading=h.e4h3tc91ger2)

[6.3 Tools 17](#_heading=h.usksc6dwpdu0)

# **1.0 Introduction**

The Software Project Management Plan outlines the scope, timeline, cost estimates, risks, and project management approach for the development of a mobile application that tracks and smooths virtual writing drawn by a finger using LED-equipped gloves. This is achieved through advanced image processing and machine learning techniques.

The primary objective of this project is to provide accurate real-time recognition and stabilization of hand gestures, enabling smooth virtual writing. The mobile application will also be adaptable for use in VR/AR environments. The successful implementation will result in a robust and scalable system that delivers seamless gesture control and enhances user interaction across multiple platforms.

## Problem statement

Patients with Parkinson’s disease often face challenges with tremors and instability in hand movements, making it difficult to interact with technology through traditional gesture-based interfaces. Current solutions for capturing and tracking hand gestures lack the precision and smoothness necessary for this population, resulting in inaccurate recognition and frustrating user experiences.

There is a need for a system that can stabilize and smooth finger movements, particularly for virtual writing, to provide a more intuitive and accessible interface. The absence of a robust solution that leverages advanced image processing and machine learning techniques to correct these tremors and refine gestures in real time limits the effectiveness of existing technologies. Additionally, this system must integrate seamlessly into VR/AR applications and other interactive platforms, such as smart home devices, to enhance the independence and quality of life for patients with Parkinson’s disease.

## Project scope

This project focuses on developing a mobile application that captures, stabilizes, and smooths virtual hand movements for patients with Parkinson’s disease, specifically for virtual writing. Key features include:

* **Gesture Tracking and Stabilization**: Real-time tracking of finger movements using LED-equipped gloves and a rolling shutter camera. Image processing techniques will be applied to stabilize and smooth gestures, minimizing tremor effects.
* **Machine Learning for Gesture Smoothing**: A machine learning module applies a hybrid LSTM-CNN model processes the data for accurate spatial and temporal gesture recognition, producing smooth 3D trajectories in real time.
* **VR/AR Integration**: The system will support virtual reality (VR) and augmented reality (AR) applications, allowing users to visualize and interact with their refined hand movements.

### 1.2.1 Inclusions

Modules for gesture tracking using LED-equipped gloves and a camera, image processing, machine learning, and VR/AR integration.

### **1.2.2 Exclusions**

The project does not include integration with third-party applications beyond VR/AR environments.

## **1.3 Major software functions**

The software functions will be divided into the following major modules, with key processes for each:

* **Mobile Application:** Serves as the main user interface, allowing the user to start and stop gesture tracking with a button and manage the setup of the LED gloves and camera
* **LED Gloves:** Captures real-time hand movements during virtual writing.
* **Image Processing Module:** Processes the video data from the camera, tracks hand movements, and stabilizes them by reducing noise using bicubic interpolation and Kalman filtering, preparing the data for the next stage.
* **Machine Learning Module:** Further smooths and refines the gesture data using a hybrid LSTM-CNN model, ensuring accuracy and consistency.
* **3D Simulation Module:** Visualizes the smoothed gestures in real-time, depicting the virtual writing in a 3D space.
* **VR/AR Integration:** Allows the user to visualize their gestures or virtual writing in virtual reality or augmented reality environments, providing an immersive experience.

## **1.4 Performance/Behavior constraints**

The software must meet the following performance and behavior constraints:

* **Real-Time Response for Gesture Tracking and Processing:** The system must process input frames and generate the 3D path of gestures in real-time, with minimal latency. It is acceptable if some operations, ie. Post-Processing operations (smoothing) take a few milliseconds, but the overall experience must feel smooth, responsive and indistinguishable from their own hand motions.
* **Continuous Frame Processing:** The system should handle a continuous stream of 60Hz image frames without missing or delaying frame processing, this will ensure smooth tracking of the gestures.
* **Efficient Pre-Processing and Feature Extraction:** Pre-Processing (sharpening, denoising etc.) and feature extraction must be performed efficiently to maintain the real-time tracking capabilities of the system.
* **Scalability:** The system should be able to handle various hardware configurations (smartphones with varying processing power) without compromising performance. It should be optimized to perform the gesture tracking and smoothing on any shutter camera integrated mobile device without any noticeable delay.
* **Accuracy in 3D Path Reconstruction:** The software should ensure accurate reconstruction of the LED’s 3D path, providing precise X, Y, Z, and Time(T) for further use.
* **Proper Smoothing of the Path:** The software must accurately smooth the constructed path in an time efficient manner thereby improving its readability by the User without significantly altering the Users directive.
* **User-Friendly Interface:** The mobile application interface should be intuitive and easy to use, allowing users to start and stop gesture recording effortlessly, with easily understood feedback.
* **Privacy Protection:** The system should effectively filter out irrelevant scene data to ensure user privacy while maintaining a high degree of accuracy in tracking the LED-equipped gloves.

## **1.5 Management and technical constraints**

### 1.5.1 Management constraints

* The completion of the project will be done by 4 people.
* The software planning and scoping must be completed by November 17th, 2024.

### 1.5.2 Technical constraints

* This software project must utilize programming languages that support advanced image processing and machine learning, like Python.
* The system will rely on real-time data capture and must ensure efficient handling of gesture tracking.
* A cloud-based database using Firebase will be used to manage user and session data.

# 2.0 Project Estimates

This section provides cost, effort, and time estimates for the Adaptive HCI – 3D Gesture Tracking Using LED-Equipped Gloves project, including estimates based on historical data, lines of code, function points, and tasks/processes.

## 2.1 Historical data used for estimates

The estimates in this project are based on prior experience with software development projects that involve advanced image processing, machine learning, and real-time gesture tracking. The team has worked on similar projects involving mobile app development and integration of machine learning algorithms for real-time data processing.

## 2.2 Estimation techniques applied and results

This project’s estimates were derived using three primary estimation techniques: **Lines of Code (LOC)**, **Function Point Analysis (FPA)**, and **Task/Process-Based Estimation**. Below are the techniques, assumptions, and corresponding estimates for each.

### 2.2.1 Line-of Code Estimate

To estimate the total lines of code required for the project, the functionality of the software was broken down into core components, and the code required for each function was approximated. The estimate is based on similar projects and adjusted for this project's specific needs, including image processing, machine learning, and 3D visualization.

| Table 1: Lines of Code Estimate Breakdown | |
| --- | --- |
| Function | Estimated Lines of Code |
| Main functionality (gesture tracking, UI) | 500 |
| Image processing module | 500 |
| Machine learning module | 1,000 |
| 3D visualization and VR/AR integration | 750 |
| Mobile app development (user interface, etc.) | 600 |
| Total: | 3,350 |

Using the total of **7,000 lines of code** and assuming **14 lines of code per developer per 8-hour day**, the project will take an estimated **240, 8-hour days** or **1920 person hours** for code development alone. However, this estimate does not account for testing, documentation, and integration, which are addressed in the task-based estimate below.

### **2.2.2 Function Point Estimate**

The project’s functionality was divided into external inputs, outputs, inquiries, interface files, and internal logical files, with complexity factors assigned based on the expected difficulty of implementation.

* **External Inputs (EI):** User gesture data from LED-equipped gloves, camera input for hand tracking, and user commands from the mobile application.
* **External Outputs (EO):** 3D visualization of smoothed gestures, real-time feedback of stabilized movements on the mobile application, and visual output for VR/AR environments.
* **External Inquiries (EQ):** Querying gesture data for analysis or correction, and status checks on gesture tracking progress and stability.
* **External Interface Files (EIF):** Temporary storage of captured gesture data for processing by the machine learning module, and data communication between the image processing and machine learning modules.
* **Internal Logical Files (ILF):** User session data, gesture records, calibration data for LED gloves and camera, and model training data for refining gesture smoothing accuracy.

| **Table 1: Component complexity factors** | | | |
| --- | --- | --- | --- |
| Component | Complexity | | |
| Low | Average | High |
| EI | 3 | 4 | 6 |
| EO | 4 | 5 | 7 |
| EQ | 3 | 5 | 6 |
| EIF | 5 | 7 | 10 |
| ILF | 7 | 10 | 15 |

For the functions within the project EI and EIF are average complexity and EQ, EO, EQ and ILF are high complexity.

| Table 2: Project function quantity and complexity. | | |
| --- | --- | --- |
| Category | Quantity | Complexity factor |
| EI | 5 | 4 |
| EO | 5 | 7 |
| EQ | 2 | 6 |
| EIF | 1 | 7 |
| ILF | 5 | 15 |

Assuming **8 hours per function point**, the total estimate for the project is **1,192 person hours**.

### **2.2.3 Tasks Estimate**

The tasks required to complete the project scope and estimated time to complete in hours are as follows:

| **Table 3: Task-Based Estimate** | |
| --- | --- |
| **Task** | **Estimated Hours** |
| Software project management plan | 15 |
| Requirements gathering and analysis | 30 |
| Design phase (architecture and diagrams) | 50 |
| Image processing development | 150 |
| Machine learning model development | 200 |
| Mobile application development | 150 |
| VR/AR integration | 100 |
| Initial code development | 250 |
| Code reviews and refactoring | 30 |
| Unit testing | 50 |
| System testing | 50 |
| System integration testing | 25 |
| Total | 1,100 |

Based on the breakdown of tasks, the total time required to complete the project is **1,100 person hours**. This includes all phases of development, testing, and integration.

## 2.3 Estimation techniques applied and results

The reconciled estimate combines the results of the LOC, function point, and task-based estimation techniques:

* (LOC) Lines of Code Estimate: 1920 person hours (development only)
* (FPE) Function Point Estimate: 1,192 person hours
* (TBE) Task Based Estimate: 1,100 person hours

After reconciling these estimates using a weighted approach, the total project time is estimated at approximately **1,290 person hours**. This estimate gives more weight to the task-based estimate, as it reflects the complete scope of the project, including development, testing, and management activities, while also factoring in the complexity indicated by the function point and lines of code estimates.

## 2.4 Project Resources

To complete this project, the following resources are required:

* **People**: Four core team members, each specializing in different components of the project (image processing, machine learning, mobile app development, and 3D visualization).
* **Hardware**: Smartphones with rolling shutter cameras, LED-equipped gloves for gesture tracking, and computers for development and testing.
* **Software**:
  + - **Development Tools**: Python, OpenCV, TensorFlow (or PyTorch), NumPy for data manipulation, Android Studio for mobile app development.
    - **Version Control and Project Management**: GitHub for code management, Hive for task management.
    - **Machine Learning**: Libraries such as TensorFlow and SciPy for implementing ML algorithms.
    - **Image Processing**: OpenCV for real-time processing of camera data.
    - **VR/AR Libraries**: Unity3D or Unreal Engine for VR/AR visualization.
* **Tools**:
  + - **Project Management:** Hive for task planning and scheduling.
    - **Documentation and Sharing**: Google Drive for document collaboration and sharing.
    - **Communication:** Discord and Zoom for regular meetings and discussions.

# 3.0 Risk Management

## **3.1 Project Risk Table**

| **Table 6: Project Risk Table** | | | | | |
| --- | --- | --- | --- | --- | --- |
| Description | Probability | Impact severity | Impact | Mitigation Plan | Contingency Plan |
| Technical requirements outside team experience | Low | Med | Potential design failures or issues with hardware/software integration | Early planning and requirement gathering to identify skill gaps early on | Seek external help, such as consultants or additional resources with relevant expertise |
| Inconsistent LED tracking due to ambient lighting | Med | High | Difficulty in accurately tracking LED movements, leading to incorrect data | Test under different lighting conditions; enhance image processing techniques (denoising, threshold) | Adjust LED brightness or use more advanced sensors to compensate for poor lighting conditions |
| Real-time processing performance limitations | Med | High | Slow response times may affect gesture tracking accuracy and user experience | Optimize code for performance; prioritize hardware capabilities in app development | Simplify features to reduce computational load or upgrade to more powerful hardware if possible |
| Privacy filtering failure | Low | High | Failure to remove irrelevant data could lead to privacy concerns and data leakage | Implement robust filtering algorithms and conduct regular privacy testing | Roll back feature until security improvements are made |
| Misalignment between image processing and machine learning | Med | Med | Data processed by image processing module may not be compatible with the ML module | Define strict data formats for communication between modules | Add a real-time debugging layer between the two modules to catch and adjust data discrepancies |
| Insufficient training data for machine learning | Med | Med | Machine learning models might not perform well due to insufficient or poor-quality data | Collect/Create diverse data sets early, including various lighting and movement scenarios | Supplement with synthetic data generation or manual labeling to expand dataset |
| Hardware failure or incompatibility | Low | High | Smartphone cameras may not function as expected, causing system interruptions | Conduct extensive hardware testing during initial phases | Use backup hardware or simulate inputs to keep development on track |
| Changes in hardware specifications during development | Med | High | Significant changes may force redesigns, delaying the project | Continuous monitoring of hardware compatibility with the app | Work with minimal specs to keep the app functional with basic hardware; plan updates accordingly |
| Data loss or corruption during frame capture | Low | Med | Loss of critical gesture data during frame capture could reduce tracking accuracy | Implement data validation checks and automatic re-capture protocols | Allow re-capture or redundant data logging to mitigate data loss |
| Unclear or delayed requirements from stakeholders | Med | Med | Lack of clarity may lead to rework or delays | Frequent communication with stakeholders; clearly documented requirements | Adjust scope or timeline based on new requirements |
| Machine learning accuracy does not meet expectations | High | Med | Gesture recognition may not be as accurate as expected, affecting end-user experience | Implement rigorous testing early in development and adjust models | Refine ML models, incorporate user feedback, and simplify gestures if necessary |
| Inadequate pre-processing performance | Med | High | Poor pre-processing may lead to inaccurate feature extraction and gesture tracking | Optimize pre-processing algorithms and conduct thorough testing in various environments | Simplify image processing methods if performance issues arise |
| Security vulnerabilities in mobile app | Low | Med | User data may be compromised during gesture recording | Conduct regular security assessments and ensure proper encryption and data handling | Release patches to fix vulnerabilities; notify users promptly |

## **3.2 Overview of Risk Mitigation, Monitoring, Management**

Twice weekly the status of all identified risks must be reviewed by project management. In these meetings updates will be presented and the current status of each risk updated. Risks will be tracked using green, yellow, and red statuses. As risks become more likely to happen the mitigation plan must be reviewed. In this meeting the team will determine when contingency plans should be implemented.

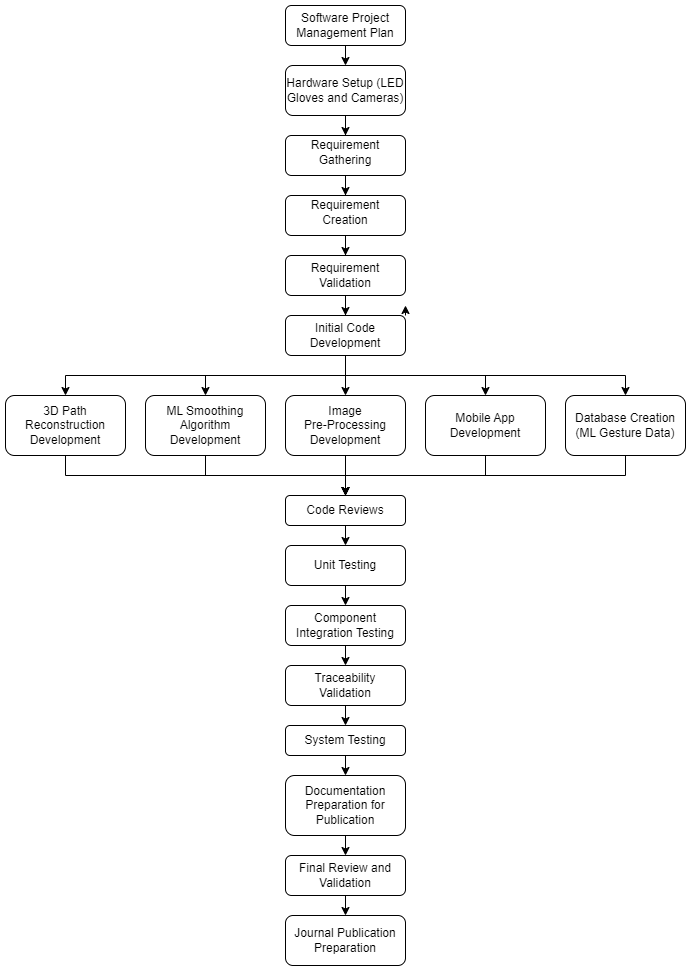
# 4.0 Project Schedule

## **4.1 Project task set**

* **Software Project Management Plan:** Develop a detailed plan that outlines the project's objectives, scope, schedule, resources, and risk management. This plan will guide the execution of the project as a proof of concept aimed at journal publication.
* **Hardware Setup (LED Gloves and Rolling Shutter Cameras):** Ensure that the LED-equipped gloves and rolling shutter cameras are functional. Set up the hardware in a controlled environment for testing and development.
* **Requirement Gathering:** Collect detailed technical and functional requirements focused on achieving real-time gesture tracking and smoothing. These will define the proof of concept without needing to address real-world deployment concerns.
* **Requirement Creation:** Document the gathered requirements as clear functional specifications for development. Focus on key features like gesture tracking accuracy, path smoothing, and privacy filtering.
* **Requirement Validation:** Validate that the requirements are well-defined, accurate, and technically feasible for the project, ensuring alignment with the project’s goals of publication and proof of concept.
* **Initial Code Development:** Begin developing the core components, focusing on image pre-processing (sharpening, denoising, edge detection) and feature extraction (LED position and depth). This is aimed at demonstrating the feasibility of the technology.
* **Path Reconstruction Algorithm Development:** Develop algorithms to track the LED’s 3D path over time using NumPy, ensuring the concept is technically viable for further exploration in academic research.
* **ML Smoothing Algorithm Development:** Implement gesture smoothing algorithms (Kalman Filters, moving average filters, spline interpolation) to demonstrate the effectiveness of tracking gestures with high accuracy.
* **Mobile App Development:** Create a basic interface for recording and displaying gestures, ensuring the app can demonstrate the concept in a research setting. Usability features for end-users are not a priority.
* **Code Reviews:** Conduct regular code reviews to ensure code quality and alignment with the project’s research objectives, promoting clean and functional code that can be shared in academic settings.
* **Unit Testing:** Perform unit testing on each component, such as pre-processing, feature extraction, path reconstruction, and gesture smoothing, to verify their functionality within the concept.
* **Component Integration Testing:** Test the integration of hardware and software components, including LED gloves, cameras, and real-time processing modules, to ensure they work together smoothly.
* **Machine Learning Integration:** Implement machine learning models to enhance the gesture tracking system, focusing on demonstrating how ML improves the tracking accuracy and smoothing.
* **Database Creation (Data Storage for Gesture Data):** Set up data storage using NumPy for gesture data (X, Y, Z, T) to support further analysis and research publications. The focus is on storing data for experimentation and validation.
* **Traceability Validation:** Ensure that each requirement is traceable to the corresponding code and design elements to maintain alignment between the original concept and the final implementation for academic publication.
* **System Testing:** Perform system-wide tests to confirm that all components function correctly together, focusing on proving the technology works under controlled conditions suitable for journal publication.
* **Documentation Preparation for Publication:** Write detailed documentation of the project, including technical methodologies, results, and findings, ensuring it is ready for academic review and publication.
* **Final Review and Validation:** Conduct a final review to ensure that all project objectives have been met and that the proof of concept is solid. Validate with the project team and academic advisors before preparing for publication.
* **Journal Publication Preparation:** Prepare the project results, findings, and supporting documentation for submission to relevant academic journals. The focus is on presenting the concept and results effectively for peer review.

## 4.2 Task network

Task network is shown below using vertical from top to bottom over time. Horizontally aligned tasks are performed at the same time.

****

## 4.3 Timeline chart

The timeline chart illustrates the schedule for the project, showing when each task is planned to start and finish. **It includes a timeline for the prototype.**

| **Table 7: Project Task View** | | | |
| --- | --- | --- | --- |
| Task | Duration (Business Hour) | Start Date | End Date |
| Software Project Management Plan | 40 | October 21st 2024 | October 23rd 2024 |
| Hardware Setup (LED Gloves and Cameras) | 60 | October 14th 2024 | December 10th 2024 |
| Requirement Gathering | 50 | September 4th 2024 | September 11th 2024 |
| Requirement Creation | 40 | September 11th 2024 | September 25th 2024 |
| Requirement Validation | 40 | September 25th 2024 | October 2nd 2024 |
| Initial Code Development | 600 | October 9th 2024 | October 21st 2024 |
| Path Reconstruction Algorithm Development | 100 | October 17th 2024 | December 10th 2024 |
| Smoothing Algorithm Implementation | 120 | October 17th 2024 | December 10th 2024 |
| Mobile App Development | 200 | October 9th 2024 | December 10th 2024 |
| Code Reviews | 100 | December 10th 2024 | December 12th 2024 |
| Unit Testing | 200 | December 12th 2024 | December 15th 2024 |
| System Integration Testing | 200 | December 15th 2024 | December 17th 2024 |
| Machine Learning Integration | 200 | October 17th 2024 | December 10th 2024 |
| Database Creation (Gesture Data) | 80 | October 31st 2024 | December 10th 2024 |
| Traceability Validation | 60 | October 2nd 2024 | December 18th 2024 |
| System Testing | 150 | December 17th 2024 | December 18th 2024 |
| Documentation Preparation for Publication | 120 | December 10th 2024 | December 18th 2024 |
| Final Review and Validation | 80 | December 17th 2024 | December 18th 2024 |
| Journal Publication Preparation | 60 | September 4th 2024 | December 18th 2024 |
| Total | 2450 | September 4th 2024 | December 18th 2024 |

# 5.0 Staff Organization

## **5.1 Team structure**

The team consists of four members, each with their own responsibilities. Tasks are divided among the team, but all members work closely together, and cross-checking is implemented to ensure collaborative decision-making.

**Members and roles are as follows:**

* **Soham Naik**: Responsible for **Image Processing**, including pre-processing, feature extraction, privacy filtering, and image processing smoothing. Soham ensures that the system captures and processes gestures efficiently, delivering high-quality input data for 3D reconstruction and machine learning modules.
* **Deniz K. Acikbas**: In charge of **Front-End Mobile App Development** and **Hardware Integration**. Deniz manages the development of the user interface and handles the integration of the camera and LED-equipped gloves with the app. Deniz also works closely on camera feed acquisition and rolling shutter handling.
* **Alan Raj**: Oversees **3D Visualization**. Alan focuses on converting the extracted gesture data into a meaningful 3D representation. This includes working with 3D path reconstruction and visualization tools to create an intuitive display for the tracked gestures.
* **Zaynab Mourtada**: Responsible for **Machine Learning-Based Smoothing**. Zaynab leads the efforts in designing and implementing machine learning models that smooth gesture paths and enhance real-time tracking accuracy. Zaynab’s models are key to refining the data output for further use in healthcare, AR/VR, and other applications.

**Shared Responsibilities**:

* **Project Validation**: All team members are equally involved in validating the project against the functional and non-functional requirements. This includes iterative testing, quality assurance, and making sure the project meets academic and journal publication standards.
* **System Integration**: The team collectively ensures that all software and hardware components work together seamlessly. This requires continuous collaboration across the different technical modules.

## 5.2 Management reporting and communication

The team employs the following communication and management structures to ensure efficient collaboration and project progress tracking:

* **High-Level Product Discussion**: Major decisions and high-level discussions take place via **Discord** or **Zoom** calls, either internally among team members or with the client. After each meeting, detailed summaries and action items are documented in **Hive** for project tracking and stored in **Google Drive** for reference.
* **Day-to-Day Communication**: For short, informal communication and real-time group discussions, **Discord** remains the primary messaging platform. This allows for quick, on-the-fly collaboration across the team.
* **Project Management**: **Hive** is used for managing the project’s timeline, tasks, and assignments. This platform tracks project progress, assigns tasks, and holds all key deliverables to ensure the project remains on schedule.
* **Code Management**: All development work, including code commits, version control, and reviews, is managed through **GitHub**. The platform also serves as the repository for the mobile app, image processing code, and 3D visualization modules, ensuring seamless integration and collaboration.
* **Document Sharing**: Project documentation, class requirements, and non-code-related files are stored on **Google Drive** and **Dropbox**. This ensures that all team members have access to the latest documents and updates, with clear versioning for reference.
* **Progress Reporting**: Project progress is tracked in **Hive** and reported through summaries during weekly team meetings. Important milestones or issues are communicated through email, and regular updates are provided to ensure visibility into key decisions and progress for all stakeholders.

# 6.0 Tracking and Control Mechanisms

## 6.1 Quality assurance and control

The team will utilize a combination of tools and methodologies to ensure that all code meets the functional requirements and expectations for this project. These include:

* **Unit Testing**: We will employ unit testing frameworks (such as PyTest or JUnit) to ensure individual components (e.g., image processing, gesture extraction) behave as expected. Mock objects will be used where necessary to isolate functionality and avoid dependency on the full stack.
* **Integration Testing**: We will test the integration between the mobile app, image processing, and machine learning modules. Integration tests will include real-time gesture capture and processing, ensuring that these systems work in concert.
* **GUI Testing**: Automated front-end testing using tools like **Selenium** for the mobile app interface will be implemented to validate basic UI interactions, such as gesture recording and real-time feedback.
* **Verification and Validation against Requirements**: Each module (image processing, gesture smoothing, 3D path reconstruction) will be verified against project specifications. This verification will be tracked in GitHub issues under dedicated milestones for each functionality. The output data will be validated to ensure accuracy, and feedback from stakeholders will be considered for improvement.
* **Continuous Integration (CI)**: All tests, including unit and integration tests, will be run in our CI pipeline on **GitHub Actions**. This ensures that any new code commits maintain system functionality and meet the required coding standards. Any failed tests will be flagged and reported for immediate resolution.
* **Staging Environment**: A staging instance of the app will be regularly updated to test features as they are developed. The team will use this environment to identify any issues and ensure that requirements are met before moving forward.

## 6.2 Change management and control

Our team separates two classes of tracked changes:

* **Git for Source Code Management (SCM)**: All source code (for image processing, machine learning, app development, etc.) will be managed via **GitHub**. We will use the **Git flow** branching model, where feature branches are merged into the main branch only after passing review and tests. Tags will mark major development milestones and deployable builds.
* **Google Drive for Document Management**: For non-code files (e.g., design documents, reports), **Google Drive** will be used for version tracking. Files will follow a versioning scheme (Draft, Reviewed, Final) and be archived systematically to ensure traceability.

## 6.3 Tools

The following tools will be employed across the project lifecycle:

* **GitHub (SCM)**:
  + **Source Code Management**: Tracks code changes and enforces version control.
  + **CI Pipeline (GitHub Actions)**: Runs automated tests, checks for code quality, and ensures that no new bugs are introduced.
  + **Code Reviews**: The pull request feature will be used to review and approve code before merging into the main branch. This ensures team-wide visibility and quality control.
  + **Issue Tracking**: Issues represent tasks or bugs to be addressed. Each issue is assigned to a milestone and will be tagged to indicate its status (e.g., "In Progress," "Completed").
  + **Continuous Deployment**: Later in the project, the CI/CD pipeline will also facilitate automated deployment to the staging environment.
* **Hive (Project Management)**:
  + **Task Tracking**: Task progress will be tracked in Hive, using milestones and due dates to ensure timely delivery.
  + **Issue Management**: Hive will also act as a system for tracking high-level issues and project management tasks, complementing GitHub’s issue tracking for code-related tasks.
* **Google Drive (Document Management System)**:
  + Tracks documents that aren’t code-related (e.g., project scope, design reports).
  + Follows a structured versioning system for drafts, reviewed documents, and finalized versions.
* **Discord (Communication)**:
  + The team will use Discord for day-to-day communication, while formal decisions will be documented in Hive and meeting summaries stored in Google Drive.

These tools collectively ensure that code is consistently reviewed, tested, and maintained, while also providing efficient management of tasks and project documents.

| **Review and Signoff of the SPMP** | | | |
| --- | --- | --- | --- |
| **Name** | **Project Team Role** | **Signature** | **Date** |
| Soham Naik | Lead Computer Vision & Image Processing Engineer | Soham Naik | 12/10/2024 |
| Deniz Acikbas | Lead Mobile App & VR/AR Engineer | Deniz Acikbas | 12/10/2024 |
| Zaynab Mourtada | Lead Machine Learning Engineer | Alan Raj | 12/10/2024 |
| Alan Raj | Lead 3D Visualizer Engineer | Alan Raj | 12/10/2024 |
| Xiao Zhang | Client | Xiao Zhang | 12/10/2024 |