Version 1.3

Requirements Document

Date: 04/29/2025

HapTech

Project Sponsor: Dr. Reza Razavian

Project Mentor: Veerendernath Surendernath Komala

Team Members: Landon Coonrod, Matthew Gardner, Peter Hilbert, Karissa Smallwood

Accepted as baseline requirements for the project:

Team	**Client**

Table of Contents

Introduction

Haptic technology is transforming the way humans interact with machines. By bridging the physical and virtual worlds, it enables users to experience force and tactile feedback, creating the sensation of physically interacting with digital objects. Haptic technology comes in various forms, from touch-sensitive surfaces that simulate textures to wearable devices that provide pressure and vibration feedback.

Graspable haptic technology has broad applications across various fields, including bomb disposal, space exploration, and medical training. Robots equipped with haptic technology either simulate or actually perform critical tasks with precision and safety. For example, medical students can practice procedures on virtual patients with haptic technology providing realistic sensations of incisions and suturing, without risking the well-being of a real person. In addition, robots are regularly used to dispose of toxic substances and explosives, which are remotely controlled using haptic technology to feel any forces the robot is exposed to in real time [3].

These applications of haptic technology are driven by the growing research area of human-robot interaction, which Dr. Reza Sharif Razavian, our project sponsor, is involved in. Dr. Razavian is an assistant professor of mechanical engineering at Northern Arizona University, where his research lab, Raz Lab, integrates robotics, neuroscience, and biomechanics. One of his key research areas involves using the Franka Research 3 (FR3) robot in human-robot interaction studies. In these studies, participants are given a task to complete in a 3D simulation, in which they move the robot's arm to control a virtual object. The robot applies forces back, simulating collisions or other movement restrictions, making the participant feel as if they are controlling a physical object. The insights from these studies are crucial for advancing applications such as motor learning, physical rehabilitation, and the design of intuitive robotic interfaces, where understanding the nuances of touch and movement can lead to more effective human-robot collaboration.

The lab's current setup to design, configure, and run these experiments with the Franka robot presents several challenges and inefficiencies, which this project aims to address. Such issues include a lack of modularity, making it difficult for researchers to set up, run, and configure experiments, and difficulty collecting and storing data from the experiments. To address these challenges, we will build a comprehensive software system that streamlines Dr. Razavian's human-robot interaction studies using the FR3 robot. The system will allow researchers to design experiments through human-readable configuration files, which an experiment operator can use to run trials through a user-friendly interface. Participants in these experiments will interact with the 3D simulations through the Franka robot with responsive force feedback to complete some defined task. A high-level overview of the system is shown in Fig. 1.

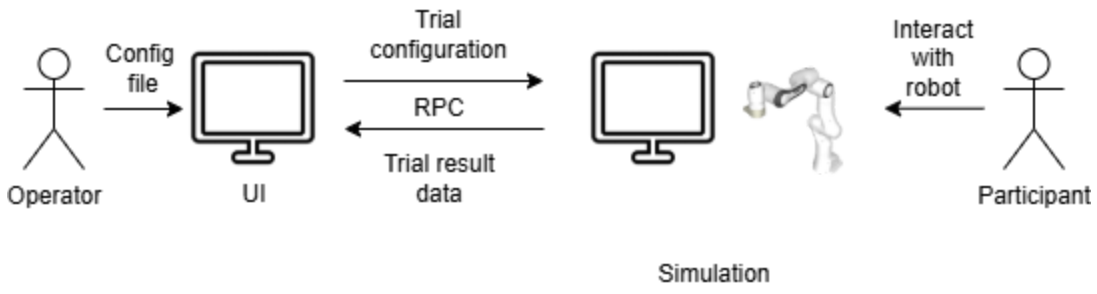


Fig. 1 - High-Level System Diagram

Problem Statement

Our client's current workflow as shown in Fig. 2, consists of the user interacting with the Franka Research 3 (FR3) robot through movement of the robotic arm. Once the user interacts with the robot, FR3 responds by sending sensor data to libfranka, the robot's control API. This data then communicates with CHAI3D and provides haptic and visual feedback to the user [2]. CHAI3D also sends control information back through a custom integration layer to libfranka, which moves the robot in response [3]. Essentially, the user's actions affect the robot causing the system to continuously update the visuals and haptic forces the user experiences.

While this system allows the user to have a basic haptic interaction, there are multiple issues with this current setup that makes it difficult for our client to conduct research. This system lacks flexibility, data handling capabilities, and user accessibility, resulting in a time-consuming and error-prone workflow.

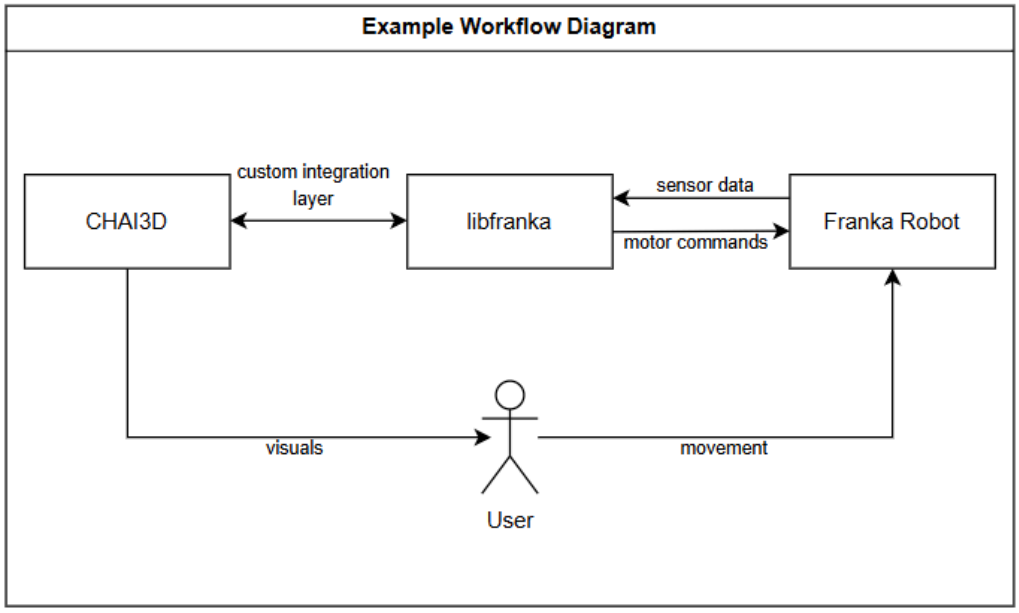


Fig. 2 - High-Level Workflow Diagram

Identified Issues:

- **No user-friendly interface:** No way for the client to create their own experiments without hardcoding, causing the setup to be slow. These experiments refer to interaction tasks where the user moves the robot arm in a 3D simulation, experiencing force feedback that mimics real-word collisions.
- **No built-in way to collect or store data:** During these experiments, the interaction data like robot position, force feedback, etc. is collected. However, our client has no way to gather or store this data for later analysis.
- **No experiment templates:** Each experiment has to be created by the client themselves, with no reusable structure to help. This means that each study, no matter the interaction task, has to be fully developed which wastes time and limits reproductivity.
- **Time-consuming process:** Manual coding each experiment as well as a lack of modularity leads to a slow and inefficient development process. This slows down the research process and limits the number of studies our client can run within a session.

Solution Vision

The proposed system for this project is a modular software platform designed to support human-robot interaction research for our client, using the Franka Research 3 robot and CHAI3D. It will allow researchers to easily design and run haptic experiments through both an intuitive user interface and terminal commands. Experiment data will be collected and stored for easy analysis. The system will ensure responsive, real-time force feedback through the robot, streamlining experiment setup and significantly improving the efficiency and accessibility of conducting research.

Key Features and Benefits:

To meet the clients requirements, the system will support the following capabilities:

- Interaction Mode:
    - Command line user interface
- Data Collection and Storage:
    - CSV files containing experiment data
    - XML files for experiment configurations
    - JSON files capturing metadata
- Enhanced Accessibility for researchers to facilitate ease of experimentation.
- Modular System Architecture to simplify future modifications and feature additions.
- Real-Time Performance ensuring accurate and timely haptic feedback.

Data and Process Flow:

- Inputs:
    - User Configuration
    - Sensor Data from Frank Robot

    - CHAI3D Parameters
- Main Processes:
    - Experiment Initialization
    - Command-line interface/Terminal Communication to Backend
    - Feedback Loop
    - Data Logging
    - Error Handling
- Data Outputs:
    - CSV Log Files
    - Visual Output
    - System Feedback

High-Level System Architecture:

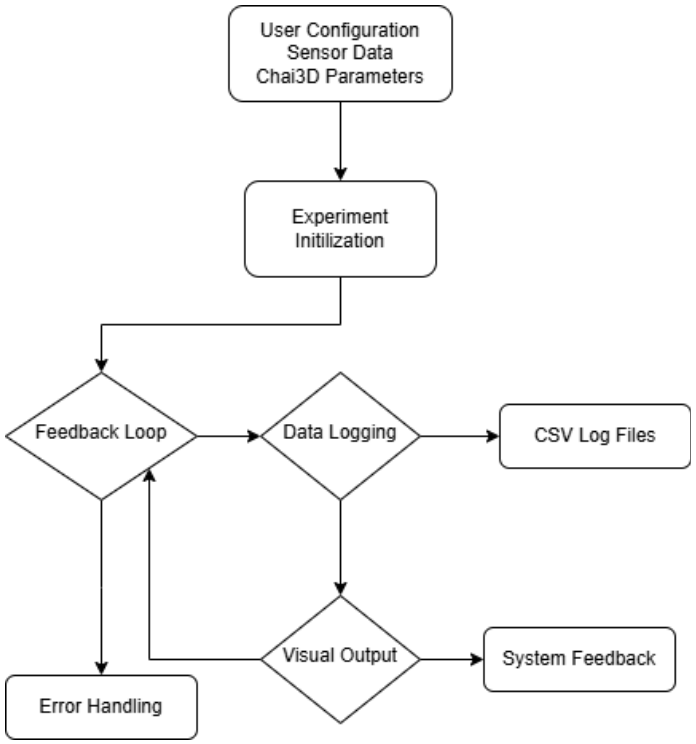


Fig. 3 - High-Level System Architecture

Discussion of Alternatives and Trade-offs:

Several alternative approaches were considered before coming to this conclusion, particularly regarding user interaction, data storage, communication protocols, and the haptic integration. This solution was chosen as it aligned best with the clients technical constraints and research goals, while still offering flexibility, maintainability, and performance in a user accessible architecture.

Project Requirements

1. Domain-Level Requirements

Domain-level requirements are the high-level requirements that outline what the system must accomplish from a user needs viewpoint. For our system, these domain-level requirements are outlined below:

D1. 3D Simulations: The system must render 3D simulations which can be interacted with via the Franka robot.

D2. Haptic feedback: The Franka robot must provide responsive and realistic force feedback from 3D simulations. The user should feel as if they are actually interacting with the virtual objects.

D3. Data collection & storage: The system should collect detailed data about the results of each experiment, and the data should be stored in an organized manner for future analysis.

D4. Experiment design and setup: Researchers must be able to easily design and modify experiments they want to run through a configuration file.

D5. Operator user interface: Researchers must be able to run experiments through a user-friendly interface.

D6. Network capabilities: The command-line interface must be able to run separately from the simulation, and possibly even on a different device. Therefore, there must be a robust communication protocol which will integrate the command-line interface and simulation components in order to provide desired functionality such as experiment control and data collection.

2. Functional Requirements

Functional requirements are the specific functions which our system must perform. We have based our detailed functional requirements on the domain-level requirements for easy traceability. Each of the following subsections breaks down the domain-level requirements into essential functionalities, and specific workflows are described in detail.

F1. 3D Simulations

- **F1.1. 3D environment:** When an experiment is running, the system shall display a graphical window containing the 3D simulation.

- **F1.2. Cursor:** Within a simulation, there shall be a 3D object that can be controlled by moving the FR3 robot's arm in three-dimensional space, which will be referred to as the cursor. The cursor serves as the user's point of interaction with the virtual environment, transmitting force feedback from simulated objects. The cursor should interact with other objects in the simulation in a realistic manner, consistent with expected physical interactions such as collision response and force feedback as determined by the CHAI3D engine.
- **F1.3. Participant Text Overlay:** Simulations shall have a text overlay indicating necessary information and instructions to the participants. For example, when an experiment first begins, there would be text instructing the participant to move the cursor to the home position in order to begin the trial. The content of such text should be able to be configured. This overlay will have the ability to be placed anywhere in the scene or have any properties depending on the configuration within the XML file.

Participant use scenario

This subsection outlines the intended experience for a participant in an experiment.

1. The participant sits at the station and grasps the FR3 robot's end-effector.
2. The 3D simulation window is visible on a nearby screen.
3. A text overlay instructs the participant to move the cursor to the home position.
4. Once the cursor reaches the home position, the trial begins.
5. The participant performs the assigned task (e.g., move the cursor to a target, apply a specific amount of force, etc.), feeling force feedback through the device.
6. When a defined end condition is met, the trial ends and a result message is shown.

F2. Haptic feedback

- **F2.1. Force feedback:** The robot shall provide force feedback during simulations. This includes simulating interactions between the cursor and other objects and any other movement restrictions imposed by the virtual 3D environment.

F3. Data collection and storage

- **F3.1. Trial metadata:** After each trial, the system shall save metadata in a JSON file. An example of this can be found in Appendix C. This will include:
    - A copy of the trial's configuration.
    - The participant ID and trial number.
    - A timestamp indicating when the trial was performed.
    - A timestamp indicating when the trial ended.
    - Whether the trial was successful or not.
    - The condition under which the trial started and ended.

- **F3.2. Interaction data:** After each trial, the system shall save data points in a file describing what happened during the trial. An example of this can be found in Appendix B. Each data point will include:
    - A timestamp.
    - The robot's state (Cartesian pose and forces).
    - Object states.
- **F3.3. Data storage:** The specified data shall be stored in an organized manner. The user should be able to choose where to store the files, otherwise, the system will automatically organize them. Generate a run folder named <subject>_<YYYY-MM-DD> and store all CSV, JSON metadata, and the copied active XML inside this folder.

F4. Experiment design and setup

- **F4.1. Configuration files:** Trials shall be specified in a configuration file, which will be written in a standard markup language such as XML. An example of this can be found in Appendix A. These configuration files will be used to construct commands to send to the server. In these files, the following will be defined:
    - **F4.1.1. Trial home position:** Each trial starts when the participant moves the cursor to the "home position." This position will be specified in the configuration file.
    - **F4.1.2. Trial conditions:** The configuration file also defines one or more start and end conditions of the trial. Conditions can be defined as a success or failure. The conditions themselves will be based on any combination of a time limit, cursor position, or cursor velocity.
    - **F4.1.3. Text overlay:** Any needed text for the simulation, as specified in F1.3, will be able to be configured, for example, text shown explaining the task to the participant.
    - **F4.1.4. Scene settings:** Various overall scene settings, such as the background color and text overlay, will be able to be customized in the configuration file.
    - **F4.1.5. Scene objects:** Configuration files will also allow for the placement of objects in the 3D environment. One such object must be designated as the cursor. Properties of objects such as position, orientation, and size will be defined here. There must be support for the following objects:
        - Box
        - Cylinder
        - Ellipsoid
        - Line
        - Sphere
        - Torus
        - Custom objects, so long as it is a CHAI3D generic object

- **F4.2. Experiment preview without collecting data:** Experiments shall be able to be previewed via their configuration file by specifying a test mode boolean flag in the XML configuration. This entails running the trial, allowing the researcher to interact with it as a real participant would. Data will not be collected from the test interactions by default, but the user will be able to override this if they wish.

Researcher use scenario

This subsection outlines the intended workflow for a researcher designing an experiment.

1. The researcher makes a copy of a template configuration file and opens it in a text editor of their choice.
2. The researcher fills in the configuration file with the needed settings, such as success and failure conditions, as well as 3D objects needed for that particular task.
3. The researcher can change the XML file to decide whether or not they want to collect data as a way to preview the scene.
4. A window opens on their computer displaying the scene.
5. The researcher makes adjustments to their configuration file as needed.
6. The experiment starts on the computer that is running the server with the robot.
7. The researcher interacts with the task through the FR3 robot to ensure it is working as intended.

F5. Operator user interface

- **F5.1. Run experiments:** The command-line interface shall allow for the user to run an experiment remotely using a configuration file. Ctrl+C will immediately end the current trial and return the system to an idle/waiting state without exiting the server side program.
- **F5.2. Preview experiments:** The user shall also have the option to run the experiment as a preview, as specified in F4.2. Previews can either be run locally on the command-line interface device or on the device with the simulation component with the robot.
- **F5.3. Display experiment status:** The command-line interface shall display the status of the current experiment in real time, including the trial success or failure, error messages, etc.
- **F5.4. Data management:** The command-line interface shall allow the user to manage and organize data files, creating directories based on participant, trial ID, etc. At the end of a trial, the system will generate a run folder named <subject>_<YYYY-MM-DD> and store all CSV, JSON metadata, and the copied active XML inside this folder.

Operator use scenario

This subsection outlines the intended workflow for an operator guiding a participant through an experiment.

1. The operator guides the participant to the robot.
2. The operator goes to their computer with the command-line interface, and opens it.
3. The operator selects the configuration file for the task.
4. The operator runs a command or selects an option to run the trial, entering the participant's ID and trial number.
5. The task loads on the simulation computer, and the operator might briefly explain the task to the participant.
6. The participant completes the trial.
7. The operator is notified shortly after that the data from the trial has been received back, and the operator saves the data to their computer.
8. Trials are repeated as necessary.

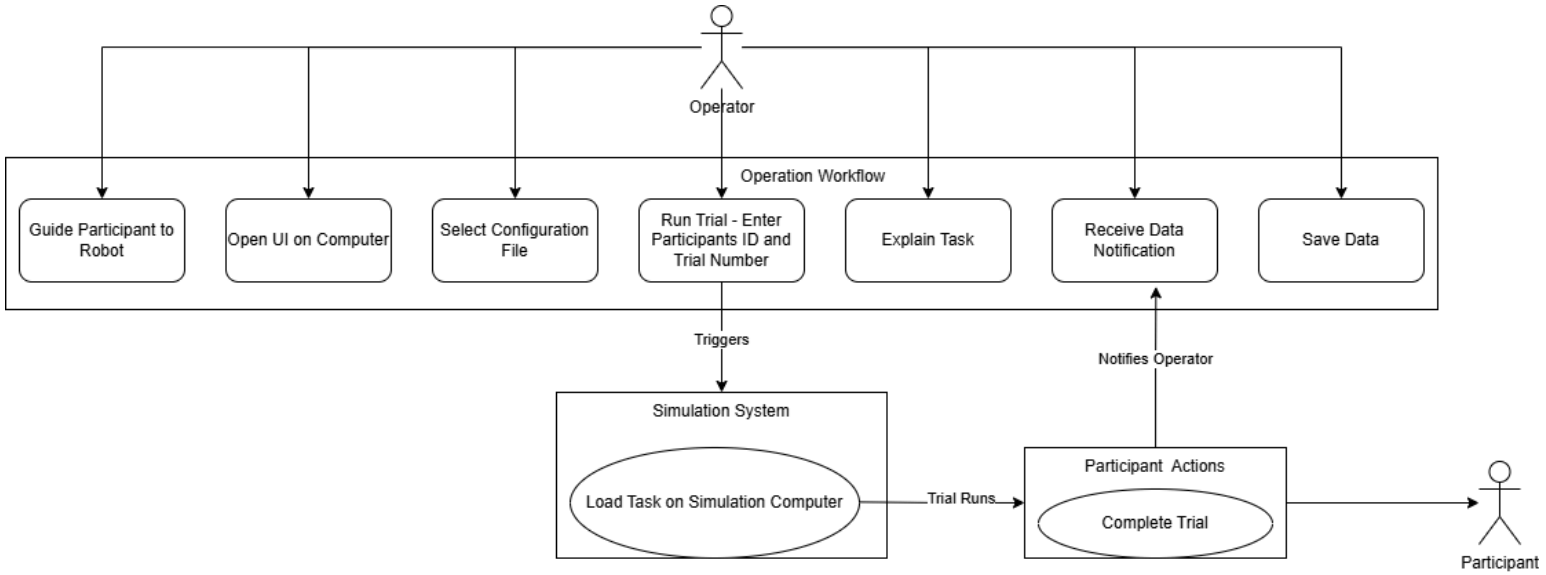


Fig. 4 - System Use Case Diagram

F6. Network capabilities

- **F6.1. Bidirectional communication:** The system shall support two-way communication between the client (command-line interface) and the server (CHAI3D/Franka robot component) over the network.
- **F6.2. Trial control:** The client shall be able to start a trial through commands sent to the server. Users shall also have the ability to end a trial from the client interface as mentioned in F5.1.

    - The client shall be able to send a command or series of commands to initiate the trial. The command(s) will include data from the configuration file provided by the user, as specified in F4.1.
    - Upon receiving commands, the server shall attempt to perform the requested actions required to construct and begin the trial. It will respond to the client with either a success or failure message.
    - When the user stops the command-line interface using Ctrl+C, the server shall handle this gracefully by resetting the scene, promptly ending the trial.
- **F6.3. Trial status feedback:** The server shall notify the frontend of trial status when appropriate.
    - Upon receiving a request to begin a trial, the server shall respond indicating whether or not the trial was initialized successfully.
    - The server shall notify the client when the participant begins the trial by moving the cursor to the home position.
    - The server shall notify the client when the trial has ended.
- **F6.4. Data transmission:** When a trial ends, the server shall send interaction data as specified by F3.2 to the client.
- **F6.5. Error handling:** The server shall notify the client of any errors that occur during the experiment process. This could include, but is not limited to, robot failure or invalid configuration data.

3. Performance Requirements

There are several performance requirements involved with this system, not only for the hardware restrictions and real time nature of the haptic simulations, but also for the sake of usability.

- **P1. Real-time feedback:** The system shall maintain a 1 kHz control loop between the haptic rendering engine and the Franka robot in order to provide smooth and realistic force feedback during interaction.
- **P2. Visual clarity:** The 3D simulation window shall render at a resolution sufficient for participants to clearly observe all objects and text overlay elements, with no reliance on zooming or window resizing.
- **P3. Visual stability:** Experiment trials shall complete without any visually disruptive glitches (e.g., frame freezing, flickering, rendering failures) that could impact user performance.
- **P4. Experiment loading time:** The system shall load and initialize any experiment within 10 seconds of the user's request, under normal operating conditions.
- **P5. Data collection performance:** Within 3 seconds of a trial ending, the client shall receive the full trial data and save it to disk under normal network and system conditions.

- **P6. System reliability:** The system must be robust during long operations; it shall be able to run at least 10 trials back-to-back without memory leaks, performance degradation, or crashes.
- **P7. Error handling and recovery:** The system shall maintain a stable state without requiring a manual reboot in the event of a recoverable failure, such as an invalid configuration or a failed object initialization.
- **P8. Efficiency and resource utilization:** The system shall effectively manage resources (CPU, GPU, and Memory) and should be optimized to prevent performance deterioration over extended use. Specifically, the following shall not be exceeded on average when running on a typical desktop computer:
    - 80% CPU utilization
    - 90% GPU utilization
    - 75% memory utilization

4. Environmental Requirements

Due to the nature of this system, there are several constraints involving hardware and software components.

Hardware Constraints

- The system must integrate with the Franka Research 3 robot.

Software Constraints

- In order to properly interface with the FR3 robot, its C++ API, libfranka, must be used.
- The CHAI3D haptic rendering engine must be used for the simulations.
- The CHAI3D and robot component must run on a Linux system with real-time capabilities.
- The command-line interface must be open source and written in Python for easy editing later.

Potential Risks

Several potential risks could impact the development and implementation of the proposed system:

Risk	Probability	Impact
Technical Complexity	High	High
Resource Availability	Medium	High
Knowledge Gaps	Medium	Medium
System Reliability	Medium	High

Risk Identification:

1. **Technical Complexity**:
    a. The integration between CHAI3D and the Franka 3 robot could be more complex than anticipated, and as we are leaving that for late in the development we may have to quickly fix errors to deliver the project.
2. **Resource Availability**:
    a. Limited access to the Franka robot in person may lead to issues for testing and development, coinciding with the above risk of a tight deadline towards the end.
3. **Knowledge Gaps**:
    a. Team members may need to spend more time than expected learning certain technologies such as CHAI3D or gRPC, affecting the projects timeline
4. **System Reliability**:
    a. Potential or unforeseen system failures or software errors that could disrupt experiment continuity may show up.

Impact Analysis:

These risks if not properly managed could significantly affect the project outcome and timeline.

- **Technical Complexity:**
    - Complex integration could delay the development timeline, comprise the performance, and reduce the overall reliability of the experiments and system.
- **Resource Availability:**
    - Access to the physical system could cause delays in critical testing phases and reduce opportunities for iterative improvement, and only give a few chances to test for bugs and issues.
- **Knowledge Gaps:**
    - Insufficient technical knowledge could slow down development and introduce errors, requiring additional time and resources for corrections.

- **System Reliability:**
    - Software failures or bugs could interrupt experiments, cause data loss, and negatively affect the user experience and trust in the system

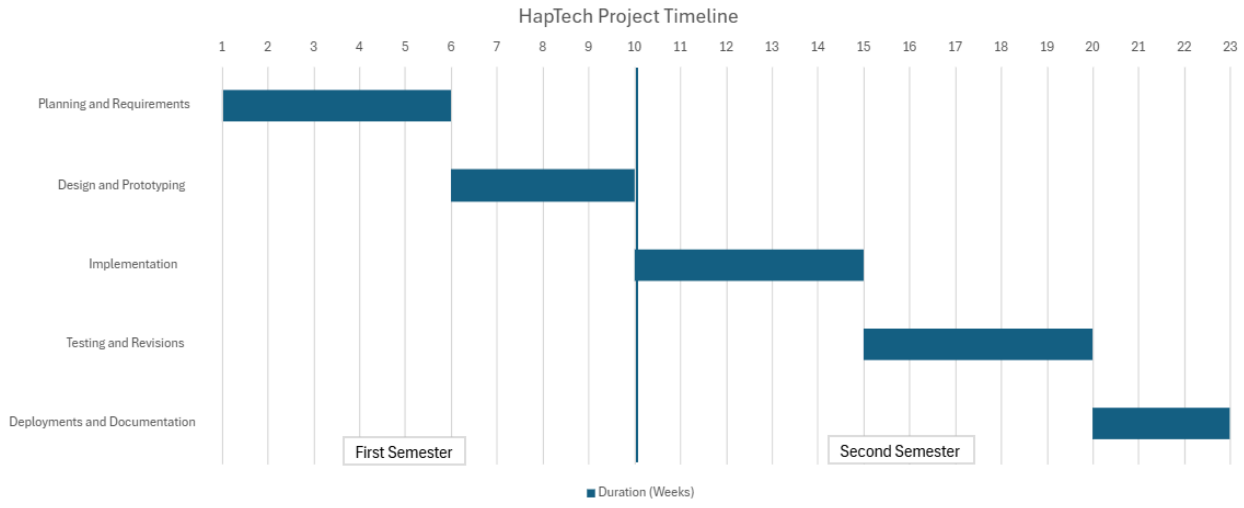
Mitigation Strategies:

To minimize these risks, the team will conduct frequent reviews of integration and current development, develop early incremental prototypes, and perform testing at each iteration. Virtual simulations and effective scheduling of lab time will address the physical needs of the project. Ongoing team training and external resources will fill knowledge gaps. Comprehensive testing practices, robust error handling, and clear user feedback will be implemented to ensure overall system reliability.

Project Plan

After careful evaluation, the HapTech team has found a sustainable project plan that aligns with our tight schedule. By using the Scrum framework, we'll break our work into sprints with each role and deliverables clearly defined to maximize efficiency and adaptability. Below is a detailed timeline, work-breakdown structure, and resource allocation that will form the roadmap for delivering our project on time and within scope.

Milestones and Timeline:



Work Breakdown Structure:

- Framework: Scrum with Sprints
    - Sprint Cycle
        - Planning: Set sprint goal and backlog

        - Daily Stand-up: 15 minute progress report
        - Execution: Develop, test, and document
        - Review: Demo the increment and gather feedback
        - Retrospective: Identify and implement process improvements
    - Roles:
        - Product Owner: Backlog prioritization and acceptance criteria
        - Scrum Master: Coaching, facilitation, roadblock removal
        - Development Team: Delivers the increment and ensures quality
    - Objective: Produce a validated, deployable increment each sprint with well defined accountability and improvement.

Resource Allocation:

- Phase 1 Feb 24 - Apr 6: Planning and Requirements
    - Research hardware/software option
    - Define system requirements and success criteria
    - Sketch high-level architecture
- Phase 2 Apr 7 - May 8: Design and Prototyping
    - Build a mock demo of the CHAI3D interface
    - Define user flows
    - Write simple scripts to simulate robot data
- Phase 3 Aug 17 - Sept 20: Implementation
    - Develop drivers for Franka 3 control
    - Build command-line interface features
    - Set up backend (TCP) for robot communication
- Phase 4 Sept 21 - Oct 25: Testing and Revisions
    - Run unit, integration, and stress tests on both robot and command-line interface
    - Log and prioritize bugs
    - Fine-tune performance and user experience
- Phase 5 Oct 26 - Nov 15: Deployment and Documentation
    - Roll out final system to the lab environment
    - Write user guides, API references, and supplemental material
    - Conduct final demo for Dr. Razavian

Conclusion

Haptic technology plays an important role in advancing human-robot interaction in enabling rehabilitation through realistic force feedback. However, our client's current system limits their ability to conduct research due to the lack of modularity, inefficiencies, and accessibilities. Improving this setup will only benefit the work on rehabilitation and robotic research.

In response, we outlined the issues within the current workflow used by our client. The key issues that we identified are: no user-friendly interface, no built-in way to manage data, no reusable experiment templates, and overall a very time consuming process. The proposed solution is a modular, command line user interface with simple data management through CSV files and a way to import configurable experiment templates. This solution provides our client with an efficient and smooth workflow that would improve their research.

As of now, we have defined domain-level and functional requirements to help guide our development in this project. We also outlined a high-level system architecture that aligns with our client needs. Following that we identified key risks and went into detail about mitigation strategies to reduce project uncertainty.

The information gathered from this document shows our passion in delivering a modular, accessible system to enable a way for more efficient research and broader usability. This can only improve the research workflow and user experience. With a well defined plan and a better understanding of challenges, we are excited to move into the design and implementation phase in the coming weeks.

Appendix A: Example Configuration XML

Below is an example of what a trial configuration XML file might look like for our system based on initial prototyping.

```
<TrialConfig>
  <!-- F4.1.1: Trial Home Position -->
  <HomePosition>
    <Position x="0.0" y="0.0" z="0.0"/>
    <Tolerance>0.01</Tolerance>
  </HomePosition>

  <!-- F4.1.2: Trial End Conditions -->
  <EndConditions>
    <Condition type="position" result="success">
      <Position x="0.1" y="0.2" z="0.3"/>
      <Tolerance>0.01</Tolerance>
    </Condition>
    <Condition type="time" value="10.0" result="failure" />
    <Condition type="velocity" result="failure">
      <MaxVelocity>0.5</MaxVelocity>
    </Condition>
  </EndConditions>

  <!-- F4.1.4 & F4.1.3: Scene Settings and Text Overlay -->
  <SceneSettings>
    <BackgroundColor r="0.2" g="0.2" b="0.2" />
    <TextOverlay>
      <Text>Move to target position (0, 0.5, 0)</Text>
      <X>400</X>
      <Y>50</Y>
     <Color r=”255” g=”255” b=”255”/>
    </TextOverlay>
  </SceneSettings>

  <!-- F4.1.5: Scene Objects -->
  <Objects>
    <Object id="cursor" type="sphere">
      <Position x="0.0" y="0.0" z="0.0"/>
      <Orientation roll="0" pitch="0" yaw="0"/>
      <Size radius="0.01"/>
      <Color r="1.0" g="0.0" b="0.0"/>
    </Object>
```

```
    <Object id="target" type="box">
      <Position x="0.1" y="0.2" z="0.3"/>
      <Orientation roll="0" pitch="0" yaw="0"/>
      <Size width="0.02" height="0.02" depth="0.02"/>
      <Color r="0.0" g="1.0" b="0.0"/>
    </Object>

    <Object id="boundary" type="cylinder">
      <Position x="0.0" y="0.0" z="0.0"/>
      <Orientation roll="90" pitch="0" yaw="0"/>
      <Size radius="0.5" height="0.01"/>
      <Color r="0.5" g="0.5" b="0.5"/>
    </Object>

    <!-- Example of a custom mesh -->
    <Object id="obstacle" type="custom">
      <File>models/obstacle.obj</File>
      <Position x="0.05" y="0.0" z="0.0"/>
      <Orientation roll="0" pitch="45" yaw="0"/>
      <Scale x="1.0" y="1.0" z="1.0"/>
    </Object>
  </Objects>
</TrialConfig>
```

Appendix B: Example CSV Output

Below is an example of the CSV output from a trial based on initial prototyping. Note that the 0s would be replaced with actual position and force values in the final product. Further, object positions are not included in this prototype, but will be in the final product.

```
t, Device Position x, y, z, Proxy Position x, y, z, Force x, y, z
0.10027092600057586,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
0.20057569000027797,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
0.3009221130005244,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
0.40123094800037507,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
. . .
```

Appendix C: Example JSON Metadata Output

Below is an example of what an output metadata JSON file might look like for our system.


```
{
    "trial_subject": "",
    "trial_id": "",
    "configuration_file": "",
    "started_at": "",
    "status_updates": [
        {
            "message": "",
            "status": 0
        }
    ],
    "trial_duration": 0.0,
    "ended_at": "",
    "configuration": {
        "start_conditions": {
            "conditions": [
                {
                    "name": "",
                    "success": true,
                    "priority": 1,
                    "parameters": [
                        {
                            "key": "",
                            "value": ""
                        }
                    ]
                }
            ]
        },
        "end_conditions": {
            "conditions": [
                {
                    "name": "",
                    "success": true,
                    "priority": 1,
                    "parameters": [
                        {
                            "key": "",
                            "value": ""
                        }
                    ]
                }
```

```
    ]
},
"hud_text": {
    "start_message": "",
    "success_message": "",
    "failure_message": ""
},
"scene_settings": {
    "background_color": {
        "r": 0.0,
        "g": 0.0,
        "b": 0.0
    },
    "camera": {
        "position": {
            "x": 0.0,
            "y": 0.0,
            "z": 0.0
        },
        "look_at": {
            "x": 0.0,
            "y": 0.0,
            "z": 0.0
        },
        "up": {
            "x": 0.0,
            "y": 0.0,
            "z": 0.0
        }
    },
    "text_overlays": []
},
"objects": [
    {
        "id": "",
        "type": "",
        "position": {
            "x": 0.0,
            "y": 0.0,
            "z": 0.0
        },
        "orientation": {
            "x": 0.0,
            "y": 0.0,
            "z": 0.0
        },
        "color": {
```

```
                    "r": 0.0,
                    "g": 0.0,
                    "b": 0.0
                },
                "params": [
                    {
                        "key": "",
                        "value": ""
                    }
                ]
            }
        ],
        "data_collection": null,
        "test": false
    }
}
```

References

[1] CHAI3D, "Open-source framework for computer haptics, visualization and interactive real-time simulation," [Online]. Available: https://www.chai3d.org/. [Accessed: Apr. 29, 2025].

[2] Franka Emika, "Franka Emika Robot – Agile collaborative robot for research and development," [Online]. Available: https://franka.de/. [Accessed: Apr. 29, 2025].

[3] Spiceworks, "What Are Haptics?," [Online]. Available: https://www.spiceworks.com/tech/tech-general/articles/what-are-haptics/. [Accessed: 26-Mar-2025].