

Insy7315 -Documentation

WIL 2025 – Quantum Developers



November 4, 2025

**Quantum Developers**

**Work Integrated Learning**

**MOVEWITHME**

**Conceptualising and planning document**

**Group Members:**

Ananta Reddy, ST10143151

Aiden Reddy, ST10290935

Shreya Dhawrajh, ST10249644

Dheyan Rambali, ST10248202

Matthew Mason, ST10403514

[**https://1drv.ms/w/c/4b361af379caf760/EbzcAZqZqr9LglvpMR2xpUcBo6QUyfwWza5I283Q3jkRmQ?e=qsA7wc**](https://1drv.ms/w/c/4b361af379caf760/EbzcAZqZqr9LglvpMR2xpUcBo6QUyfwWza5I283Q3jkRmQ?e=qsA7wc)**Sprint Doc**

[WILL MEETING MINUTES.xlsx](https://1drv.ms/x/c/4b361af379caf760/EQ0znbOir0NIqtKfn9FPB4kBI1s0q9NEBDkFpSji6WW3rQ?e=vcpsf8&nav=MTVfezRFMjE4RkIzLUI3MUMtNEU1Qi05OTk3LTY3RDNFRjc2MDFERn0)   **Meeting Minutes Doc**

**Table of Contents**

1. **Introduction**   
   1.1 Project Overview
2. **Project Management Approach**   
   2.1 Work Agreement   
   2.2 Team Roles and Responsibilities   
   2.3 Definition of Ready (DoR)   
   2.4 Roadmap (High Level Plan)   
   2.5 Change Management Strategy
3. **Requirements Engineering**   
   3.1 User Roles   
   3.3 Non-Functional Requirements   
   3.4 Security Requirements
4. **Architecture & Design**   
   4.1 Architecture Patterns

3.2 Functional Requirements   
4.2 Design Patterns   
4.3 Cloud Architecture   
4.4 Data Storage & Schema Documentation

1. **Development Practices & DevOps**   
   5.1 Development Tools and Frameworks   
   5.2 GitHub Actions Pipeline   
   5.3 Continuous Integration & Security Scanning
2. **Financial Analysis & Growth**   
   6.1 Running Costs   
   6.2 Two-Year Growth Projection
3. **Scrum & Agile Evidence**   
   7.1 Scrum Artifacts   
   7.2 Sprint Planning   
   7.3 Sprint Reviews and Retrospectives   
   7.4 Attendance Registers
4. **Testing & Evaluation**   
   8.1 Test Strategy   
   8.2 Results and Findings
5. **Appendices**   
   A. Declaration of Authenticity   
   B. Research Evidence / Diagrams   
   C. Additional Documentation
6. **User Experience Journey Map**
7. MOBSF Report
8. Work Agreement
9. Reference List
10. Annexure

**Introduction to the Project**

Our vision through our project, MoveWithMe, is to create and integrate innovative children's clothing with built-in smart sensors for monitoring and encouraging physical activity. The clothing integrates hardware and software components in a way that offers real-time feedback about the amount of activity in a child, enabling healthier lifestyles and parental supervision. The long-term objective is to encourage active play for improved development of children and inform parents and caregivers about their child's daily activity.

**Work Agreement**

Our team has established a collaborative work agreement to ensure smooth communication, accountability, and consistent progress throughout the project. All members agree to maintain professionalism, respect deadlines, and actively contribute to achieving the project’s objectives. Weekly meetings will be held to discuss progress, challenges, and upcoming tasks, while communication will primarily take place through shared project management tools and messaging platforms.

**Each member commits to:**

* Completing assigned tasks within the agreed timeline.
* Maintaining transparency and regular updates on progress.
* Collaborating effectively with other team members to ensure integration between design, frontend, and backend components.
* Upholding ethical standards and ensuring data confidentiality in all project activities.

**Team Responsibilities**

**Ananta  Project Manager**

* Oversees overall project execution and ensures milestones are met.
* Facilitates communication among team members and supervisors.
* Monitors project progress, identify risks, and implement mitigation strategies.
* Approves deliverables before submission and ensures alignment with project objectives.

**Shreya, Project Administrator**

* Maintains project documentation and meeting records.
* Manages scheduling, reporting, and coordination of resources.
* Assists with compliance tracking and supports the project manager in administrative tasks.
* Ensures effective communication between the team and stakeholders.

**Aiden, Backend Developer**

* Designs, develops, and maintains the project’s backend systems.
* Ensures secure data management and system integration.
* Conduct testing, debugging, and optimization of backend functionalities.
* Collaborates with the frontend developer to ensure seamless data flow.

**Matthew, Full Stack Developer**

* Develops both frontend and backend components of the application.
* Integrates APIs and databases for smooth system operation.
* Assists in troubleshooting and performance optimization.
* Supports deployment and version control processes.

**Dheyan,   UI/UX Designer**

* Designs are user friendly and have visually appealing interfaces.
* Conducts user research to guide design decisions.
* Create prototypes and wireframes for testing and feedback.
* Ensure consistency and accessibility across the application.

**Definition of Ready (DoR)**

Our team's Definition of Ready (DoR) ensures every user story, task, or feature is complete and unambiguous before it is accepted into a sprint. This minimises confusion, averts rework, and allows efficient collaboration across all roles.

The Definition of Ready (DoR) ensures that everything is prepared before work on the project sprint is begun. It is an explicit alignment, understanding, and readiness for all team members. The DoR is a quality gate to ensure that any task, user story, or feature entering a sprint is well defined, realistic, and achievable within the sprint duration.

**Purpose**

**The Definition of Ready is designed to:**

Ensure that all members of the team share the same understanding of the project objectives, scope, and deliverables.

Minimize delays, confusion, and rework by having readiness before commencing work.

Establish consistency and structure in planning and starting tasks for every sprint.

**Criteria**

**Project Overview and Objectives:**

The project overview, objectives, goals, and scope have been clearly defined and understood by all members of the team.

Individual project activities, for instance, the design of smart clothing prototypes and computer algorithms, have been identified and documented.

**Project Resources and Roles:**

All project team members and their responsibilities and roles have been identified and documented.

The Project Manager and Supervisors have been appointed and are aware of their role in guiding the project.

**Project Timeline and Phases:**

**The project timeline, phases, and milestones have been established and communicated to all members.**

Goals of Sprint 1, which is to concept and plan the project, have been established and clearly understood.

**Communication Channels:**

Communication channels, including face to face meetings, WhatsApp groups, Microsoft Teams, and the GitHub repository, have been set up and are active.

Minutes of meetings are recorded regularly, and progress is documented ongoing for transparency and accountability purposes**.**

**Decision Making and Conflict Resolution:**

Collaborative and equal participation decision making processes have been established.

Conflict resolution techniques, including open communication channels, responsibilities, roles, and escalation procedures, have been documented.

**Project Budget and Risk Strategies:**

The project budget has been spread across multiple phases, and detailed budgeting will be conducted at the development stage to facilitate proper utilization of resources.

Risk strategies, including how to handle technical and ethical problems, have been established and documented.

**Regulatory Compliance and Ethical Considerations:**

Regulatory compliance, for example, safety standards and ethics council approval, has been considered and planned.

Ethical considerations, for instance, confidentiality, safety of participants, and informed consent, have been addressed in the project plan.

**Technical and Functional Readiness:**

The task or feature is clearly defined, including a short description, purpose, and acceptance criteria.

All technical requirements, designs, and materials (datasets, APIs, UI mockups, etc.) are there and available for the team members to access.

Dependencies between modules, systems, or teams are identified and documented.

**Scope, Priority, and Testing Criteria:**

The scope and boundaries of each task are clear and understood by all.

The priority level and estimated effort have been decided and documented in sprint planning.

The task is verifiable, with measurable conditions for successful completion.

**Verification**

The Project Administrator (Shreya) and Project Manager (Ananta) read and confirm that all the above conditions have been met prior to including any task in the sprint backlog.

Only tasks that meet the DoR requirements are approved for active development to ensure clarity, productivity, and quality throughout the duration of the project.

**Definition of Done (DoD)**

The Definition of Done (DoD) is the shared understanding of the team about what done looks like for a deliverable, task, or feature. It ensures that all components of the MoveWithMe project meet the highest standards of quality, functionality, and compliance before they are considered done. The DoD eliminates inconsistency, responsibility, and secrecy at each development phase design and coding, testing, and delivery.

**General Completion Criteria**

**A feature, task, or sprint delivery is Done when:**

All functional requirements have been implemented and verified to be correct.

Code has been reviewed to be readable, maintainable, and project coding standards compliant.

The feature is passing unit, integration, and user acceptance testing (UAT) without major issues.

UI/UX components match approved design mock ups and meet accessibility and usability requirements.

All documentation (API references, Git commits, and technical notes) are accurate, updated, and complete.

All ethical and regulatory compliance requirements have been met, such as privacy, child protection, and safety guidelines.

 The feature or component has been implemented into the working system with no regression issues.

The Project Manager (Ananta) and Project Administrator (Shreya) have approved and vetted the deliverable.

**Hardware Deliverable, Cape Functionality**

For the MoveWithMe Smart Cape to be considered complete:

Every hardware component of the cape is in working order and communicating as it should. The power supply is solid and providing sufficient power to the Arduino and all connected components.

There is adequate communication between the:

Accelerometer and the Arduino.

LEDs and the Arduino.

Wi-Fi board and the Arduino.

SD card reader / DFPlayer and the Arduino.

The entire hardware system is functioning as one to detect movement and send data as designed.

**Software Deliverable – App Functionality**

For the mobile application to qualify as complete:

The app must be able to integrate with the API successfully, receive, and send data from the Arduino in real time.

All requested features enumerated by supervisors and customers have been developed.

The interface of the application must look like the approved design and provide a seamless user experience.

The system must provide correct and timely movement analysis in sync with the data of the smart clothing.

**Testing Requirements**

**Cape Testing:**

The cape must correctly perceive direction and motion, turn on LEDs and report to the API.

Sensor reads must be accurate, consistent, and verified against known results.

**App Testing:**

The app must effectively send and receive requests to and from the API, enabling secure two-way communication.

End-to-end testing must ensure hardware and software are functioning as a system.

**Verification**

Prior to any deliverable being labelled Done:

The Project Manager (Ananta) verifies that all items meet the DoD checklist and functional needs.

The Project Administrator (Shreya) logs and captures verification for traceability and accountability.

Only when all requirements are fulfilled, the task or feature is marked complete in the project repository.

**Roadmap (High-Level Plan)**

The team has designed a high-level roadmap as a properly structured and detailed document to guide the successful implementation of the MoveWithMe project from conceptualization through deployment. The roadmap divides the project into distinct phases with clearly defined objectives, deliverables, and checkpoints. It has constant coordination between hardware and software parts, adherence to ethical and regulation standards, and uniform development towards the final prototype and presentation.

**Phase 1: Research & Planning (Weeks 1–2)**

Objective: Establish a firm foundation for the MoveWithMe project.

Key Activities:

1. Research smart clothing technologies in place, sensor types (accelerometers, gyroscopes), and safety standards for child wearables.
2. Identify the target audience and gather requirements from potential end-users (parents, caregivers, and child development professionals).
3. Allocate responsibilities and roles to every member of the team, i.e., hardware, software, and administrative work.
4. Obtain ethical clearance and privacy considerations for collecting and storing children's activity data.
5. Develop the first project documents, including a requirements specification document, communication plan, and timeline structure.
6. Begin conceptual sketches and initial brainstorming on top concepts for clothing designs with consideration for hardware location and ergonomics.

Deliverables:

* Project proposal and ethical clearance approved.
* Project requirements documented and scope definition complete.
* First draft of hardware and software architecture overview.

**Phase 2: Design & Prototype (Weeks 3–4)**

Objective: Create and prototype concepts for the smart clothing and mobile app.

Key Activities:

1. The UI/UX Designer (Dheyan) creates initial wireframes, mock-ups, and visual prototypes of the mobile app parents will use to monitor their child's activity.
2. The Backend Developer (Aiden) and Full Stack Developer (Matthew) plan the system architecture, including the database model, communication pathway, and sensor integration through APIs.
3. Project Manager (Ananta) and Project Administrator (Shreya) arrange design review sessions and feedback loops.
4. Select and verify suitable hardware components (Arduino boards, sensors, Wi-Fi modules) to integrate.
5. Begin developing the hardware prototype design, taking into consideration comfort, flexibility, and safe sensor incorporation into fabric.

Deliverables:

Approved UI/UX design and application workflow.

First hardware prototype with first sensor layout.

First integration plan between hardware sensors and mobile software.

**Phase 3: Development (Weeks 5–8)**

Objective: Combine the building block hardware and software elements and combine both halves together to create a functional prototype.

Key Activities:

1. Implement backend functionality like data storage, API development, and real-time sensor and mobile app communication.
2. Create mobile app UI utilizing Kotlin inside Android Studio, with simplicity of data representation for parents.
3. Install and configure Arduino sensors to detect movement and respond (e.g., LED recognition of movement).
4. Conduct internal system integration to check proper communication among the hardware and software elements.
5. Begin unit testing of every module to ensure reliability and performance.
6. Schedule bi-weekly sprint reviews to discuss progress and make any adjustments required.

Deliverables:

Functional prototype demonstrating communication among the hardware and mobile application.

Functional API interfaced with hardware sensors.

Test results reported for internal validation.

**Phase 4: Testing & Refinement (Weeks 9–10)**

Goal: Validate the MoveWithMe prototype for accuracy, usability, and safety features.

Key Activities:

1. Schedule user testing sessions with children under supervision without breaching ethical guidelines and with parental consent.

1. Interval date sensor precision and reliability with comments from neonatologists and occupational therapists.

1. Perform stress testing of hardware (battery life, sensor stability) and software (API consistency, usability of user interface).

1. Gather user feedback and apply refinement to staff comfort with the hardware as well as user experience with the app.

1. Improve project documentation, including user guides, safety information, and data handling procedures.

Deliverables:

Optimized and validated smart clothing prototype.

Updated and improved mobile application.

Done documentation and test reports.

**Phase 5: Deployment & Evaluation (Weeks 11–12)**

Objective: Finish, deploy, and hand over the completed MoveWithMe project.

Key Activities:

1. Complete final preparations for final demonstration and presentation of MoveWithMe system, including live prototype testing.
2. Conduct final quality assurance checks and ensure all ethical and technical requirements are fulfilled.
3. Evaluate overall system performance, user satisfaction, and data reliability.
4. Share results, insights, and plans for future scalability (mass production, advanced analytics, and IoT connectivity).
5. Save all project reports, documentation, and code to the GitHub repository for easy access.

Deliverables:

Functional prototype and mobile app demonstration.

Final evaluation report and presentation.

Future suggestions for enhancements and production.

Sprint Duration:

* Each sprint lasts two weeks, with focused periods of development followed by review and reflection sessions.
* Sprint targets are set at the beginning of each cycle and reviewed at the end in sprint retrospectives.

**Team Availability & Coordination:**

Availability of each team member has been confirmed for the project duration.

There are daily meetings on Microsoft Teams and physically for monitoring progress, sprint planning, and resolving problems.

The roadmap accommodates academic timings to allow for level workloads and consistent productivity.

Collaborative software applications (GitHub, Google Drive, and Trello) are used for collaboration, documentation, and project management.

**Requirements**

This part gives the requirements gathered for the project, including user roles, research findings, and user stories. They were gathered during team meetings, stakeholder analysis, and research into similar systems in a bid to ensure that the software is capable of addressing the needs of the target users suitably.

**Research Summary**

To better understand the organization and its needs, the team studied in-place systems for handling project workflow, task submission, and user collaboration. We examined:

* Competition web applications for functionality, design, and data management.
* Best practices for user interfaces to offer usability and accessibility.
* Backend architecture patterns for scalability and maintenance.

The research helped guide the development of clear user roles, specified    responsibilities, and clear user stories by functionality in the system.

**User Roles**

1. Project Manager (Ananta)

* Oversee the overall progress of the project.
* Allocates tasks to team members.
* Reviews have completed work and ensured deadlines are met.
* Monitors project milestones and documentation.

1. Project Administrator (Shreya)

* Maintains documentation and meeting records.
* Organizes and tracks progress across all project phases.
* Ensures communication between all team members.
* Assists with reporting and quality control.

1. Backend Developer (Aiden)

* Develops and maintains the database and server-side logic.
* Ensures efficient data handling and system performance.
* Integrates APIs and manages data persistence.
* Works closely with the Full Stack Developer to align backend functionality with the front end.

1. Full Stack Developer (Matthew)

* Develops both client-side and server-side components.
* Connects the UI with backend logic.
* Assists in debugging and deployment.
* collaborates with the UI/UX designer for seamless implementation.

1. UI/UX Designer (Dheyan)

* Designs intuitive and accessible user interfaces.
* Creates mock-ups, wireframes, and prototypes.
* Focuses on usability and user satisfaction.
* Works with developers to ensure design feasibility.

**User Stories**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **User Story** | **Priority** | **Team Estimation** | **Sprint** | **Status** |
| Ananta | As a Project Manager, I want to assign tasks to team members so that the project progresses according to schedule. | High | 5 points | Sprint 1 | Implemented |
| Shreya | As a **Project Administrator**, I want to store meeting notes so that all members can refer to them for progress updates. | High | 5 points | Sprint 1 | Implemented |
| Aiden | As a Backend Developer, I want to design and implement a database so that all user data and tasks are securely stored. | High | 5 Points | Sprint 2 | Implemented |
| Matthew | As a **Full Stack Developer**, I want to connect the frontend UI to the backend API so that data can flow seamlessly between the two. | High | 5 Points | Sprint 2 | Implemented |
| Dheyan | As a **UI/UX Designer**, I want to create an attractive, user-friendly layout so that users can easily navigate and interact with the system. | High | 5 Points | Sprint 2 | Implemented |
| Ananta | As a **Project Manager**, I want to view overall team progress so that I can monitor performance and deadlines. | High | 5 Points | Sprint 1 | Implemented |
| Shreya | As a **Project Administrator**, I want to generate progress reports so that stakeholders can be updated regularly. | High | 5 Points | Sprint 1 | Implemented |
| Aiden/ Matthew | As a **Developer**, we want to receive notifications when tasks are updated so that we stay informed about project changes. | High | 5 Points | Sprint 2 | Implemented |
| Dheyan | As a **UI/UX Designer**, I want to gather user feedback on the interface so that future improvements can be made. | High | 5 Points | Sprint 2 | Implemented |

**Non-Functional Requirements**

The non-functional requirements define the performance and quality characteristics of the system. These requirements were gathered while discussing with the client and are utmost important to make sure that the software executes reliably, securely, and efficiently under all expected conditions.

1. Security

The system must provide basic encryption and credential protection so that user information is protected. Sensitive information such as login credentials and user accounts must be safely stored and transmitted. Access control mechanisms are implemented to prevent unauthorized access to the system.

2. Performance

The system must be well-behaved and responsive on all supported hardware. Animations must run without noticeable lag, and operations must be planned to minimize power usage to preserve battery life, especially on mobile or portable hardware.

3. Usability

The user interface must be simple, easy, and intuitive to use for all. Navigation should be straightforward with a minimal design and uniform design elements. The system must minimize the learning curve so that new users can effectively utilize it without prolonged training or documentation.

4. Reliability

The system must be extremely reliable and fault tolerant. It must be both online and offline so that users can perform crucial operations even when there is no internet. Once internet access is restored, automatic synchronizing of data must be performed.

5. Cost Efficiency

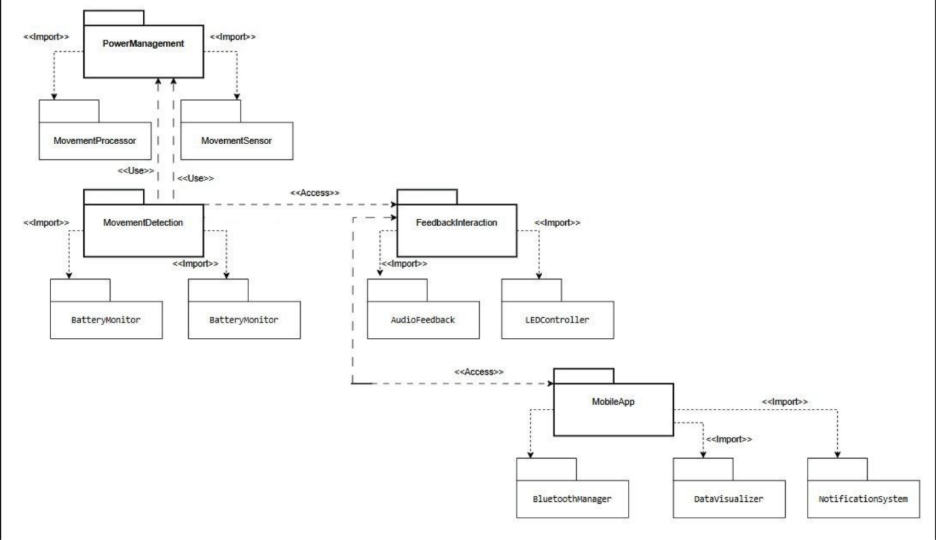
The system must also be affordable, and the development and deployment does not go above the budgetary limits laid out. The hardware to facilitate the application must not exceed a R200 budget, hence making it within reach of the client.

**Analysis Artifacts**

This overview describes the analysis that was conducted by the team to convert the requirements into formalized domain model and detailed design artifacts. Analysis followed a Domain-Driven Design (DDD) path, with the system being divided into bounded contexts so that responsibilities can be compartmentalized and integration simplified between modules.

Domain Modelling

The team identified several bounded contexts based on the functional areas of the system. Each bounded context is represented by an orderly group of components for a single purpose under the overlying architecture. The diagram below illustrates the relationship of these contexts to modules that exist in each.



**Bounded Contexts Identified:**

Power Management Context:

Responsible for managing device power consumption through features such as Movement Processor and Movement Sensor. It maintains efficient power consumption by monitoring and adjusting system activity.

Movement Detection Context:

Detects and interprets physical movement. It coordinates with Battery Monitor and Power Management to ensure movement data is managed efficiently and without unnecessary power consumption.

Feedback Interaction Context

Controls user interaction via feedback channels such as Audio Feedback and LEDController. It provides immediate responses to user actions or sensor recognitions, making user interaction more interactive.

Mobile App Context:

It is the user-facing component, communicating with Bluetooth Manager, Data Visualizer, and Notification System. It collects data from the wearable device, manages Bluetooth interactions, and shows health information to the user.

**Design Artifacts**

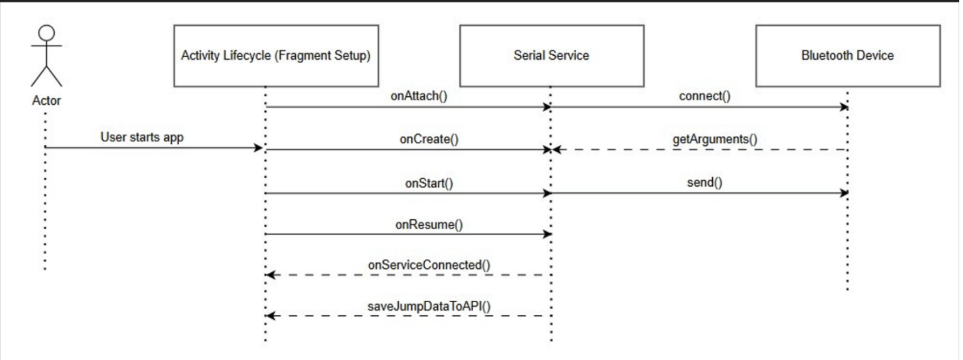
Once the domain model was defined, the team analysed component interactions to develop implementation models. The primary focus was on how the mobile application interacts with system services and external hardware devices (e.g., Bluetooth).

Figure 2 presents a Sequence Diagram depicting the flow of actions when the user starts the application. It shows how the *Activity Lifecycle*, *Serial Service*, and *Bluetooth Device* communicate during the initialization process.

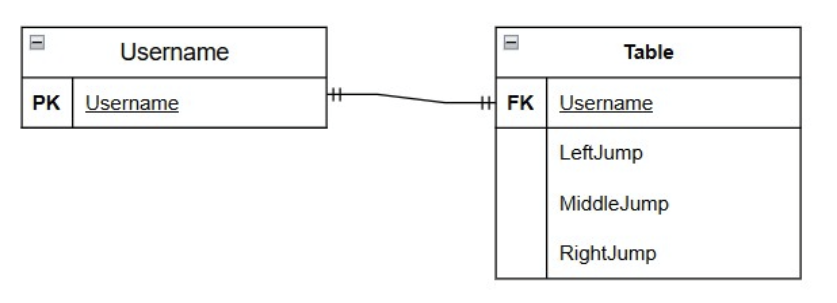
Sequence Flow Description:

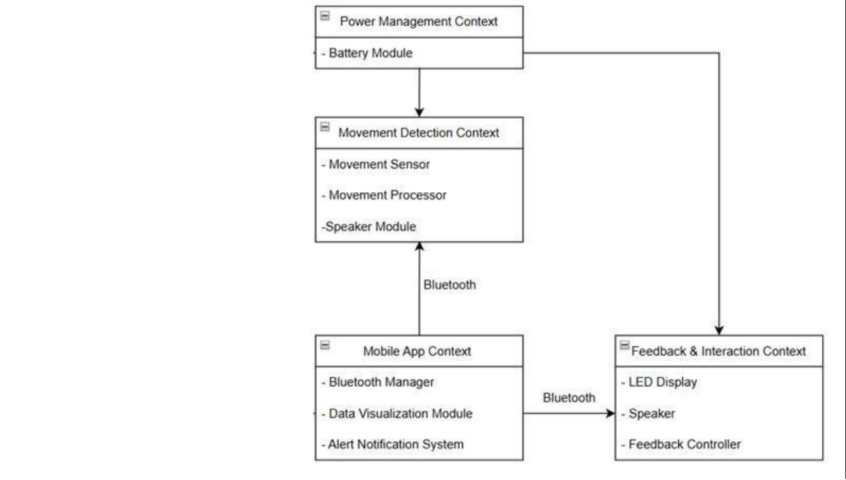
1. The user initiates the mobile application.
2. The Activity Lifecycle begins, executing methods such as onAttach(), onCreate(), onStart(), and onResume().
3. The Serial Service establishes a connection with the Bluetooth Device by invoking connect() and retrieves arguments using getArguments().
4. Once connected, the device sends data back using the send() method.
5. The service triggers the onServiceConnected() event, and the activity saves the data to the API via saveJumpDataToAPI().

This sequence illustrates the communication flow and timing between system components, providing insight into the runtime behaviour and data transfer mechanisms.

**(Figure 2)**

**Implementation Documentation**





**Security**

Security is a critical component of the system design and development process. There were various layers of protection for data, user privacy, and system integrity employed by the team. The application follows secure development practices and considers potential threats, vulnerabilities, and network-related risks.

1. Data Protection and Encryption

All sensitive user data, including login credentials and personal data, is processed securely. Passwords are hashed before they are stored, and no plaintext credentials are stored in the system. Data transferred between the client and server is encrypted through secure communication channels (such as HTTPS or encrypted APIs) to prevent interception or modification.

2. Authentication and Access Control

The system employs a role-based authentication mechanism so that protected features can be accessed only by registered users.

Session management is implemented so that user sessions expire after intervals of inactivity to mitigate the risk of unauthorized access.

Access permissions are assigned based on user roles, i.e., Administrator, Developer, or General User, for proper control of system functions.

3. Network Security

All communication between the application and backend services is secured to prevent data breaches and unauthorized data manipulation. Best practices such as firewalling, input validation, and data sanitization have been implemented to reduce the exposure to common web-based vulnerabilities like SQL injection, cross-site scripting (XSS), and man-in-the-middle (MITM) attacks.

4. Vulnerability Management

Code reviews, testing, and static code analysis were performed on a continual basis to identify and resolve vulnerabilities early in the development process. Third-party dependencies and libraries were reviewed for being up-to-date and free of known security vulnerabilities.

Furthermore, error handling mechanisms are employed for avoiding the exposure of sensitive information via system error messages.

5. Threat Mitigation

Possible risks such as unauthorized access, data leakage, and denial-of-service (DoS) attacks were identified by the team. To prevent these:

* Input validation prevents malicious script execution.
* Rate limiting and request throttling guard against brute-force or spam attacks.
* Secure file handling ensures uploaded files are validated and stored appropriately to prevent injection or malware attacks.

6. Ethical and Compliance Considerations

The project also complies with ethical standards and data privacy legislation. Users are informed of data collection, and their consent is obtained where necessary. Confidentiality and integrity of user data are prioritized at all stages of development and deployment.

7. Ongoing Monitoring

To ensure ongoing protection, processes were implemented for continuous monitoring, logging, and auditing of system activity. This offers early detection of anomalous activity, with swift response and recovery in the event of a security incident.

**DevOps**

Speed of deployment and continuous improvement are two principles that guided our team's development process. To obtain a quality and timely-deployed application, we employed a DevOps model integrating automation, testing, and deployment pipelines. This method provides consistency, reduces manual work, and assures secure delivery across the software life cycle.

1. DevOps Approach

Our team utilized a Continuous Integration and Continuous Deployment (CI/CD) strategy for guaranteeing code quality, ease in testing, and automating build. Utilizing GitHub for version control, we ensured that each code update was thoroughly tested prior to integration.

Some of the primary objectives of our DevOps setup are:

* Ensuring that each code change automatically gets tested and verified.
* Facilitating early vulnerability detection through automated security scans.
* Enabling future integration with automated deployment pipelines.

Below are the pipeline steps (as illustrated by the attached diagram):

Code Push to Repository:

Developers commit and push their code modifications to GitHub.

MobSF Security Analysis

MobSF offers full security scanning of Android applications, exposing security weaknesses before deployment. This fulfils our non-functional requirements of security, privacy, and data protection.

Report Vulnerabilities:

Security scans are executed to identify and document possible vulnerabilities within the codebase or its dependencies.

This progressive process ensures that the application is secure and best practice compliant.

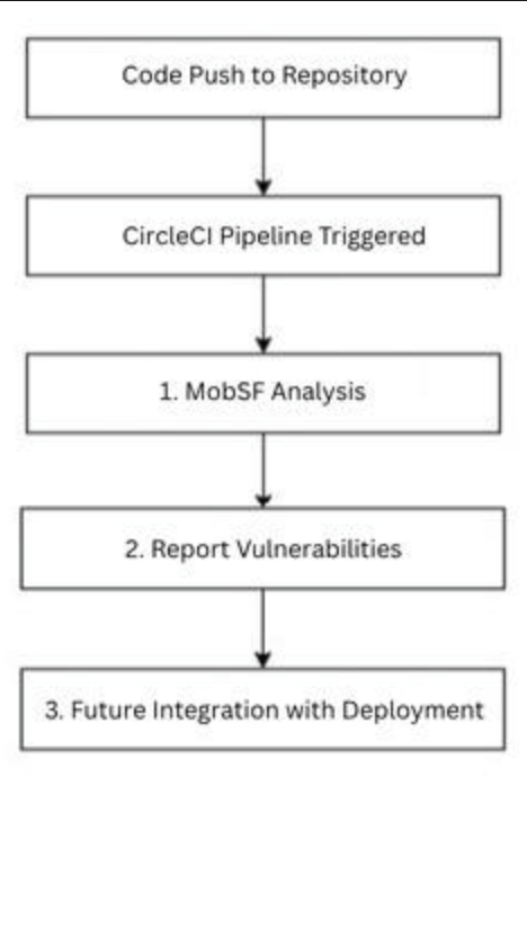
Future Integration with Deployment:

While the current pipeline is focused on continuous integration, it is designed to be easily extended to continuous deployment (CD).

Future integration will include deployment to cloud environments (e.g., Azure or AWS) on successful tests.

3. Benefits of the CI/CD Pipeline

* Implementation of this automated DevOps pipeline was beneficial in several ways:
* Quicker Delivery: Rapid iteration and testing allowed for quicker feature deployment.
* Enhanced Security: Scanning for vulnerabilities and vulnerability reports helped early threat detection.
* Scalability: The pipeline is scalable for future integration with automated deployment systems.



**Running Costs**

The MoveWithMe solution will require low but constant monthly operational costs to ensure both the intelligent wearable device hardware and phone software function at their best once deployed. According to counsel and recommendation from our client, Sarina, the projected costs aim at maintenance, gyro, and support for users, with additional scaling expected as more users adopt the solution

Monthly Running Cost Breakdown

|  |  |  |
| --- | --- | --- |
| Cost Category | Description | Estimated Monthly Cost (ZAR) |
| Miscellaneous Operational Costs | Costs for communication tools (Microsoft Teams, WhatsApp), version control (GitHub), and testing materials. | R0 |
| Hardware Components | Purchase of a new gyroscope sensor ESP32 microcontroller, USB-C cables, and switches used for testing and connectivity during development. | R200 |
| Technical Support | Providing user assistance, updating documentation, and performing remote troubleshooting. | R0 |
| App Maintenance & Updates | Regular bug fixes, Android SDK updates, feature improvements, and performance optimization. | R0 |
| Sensor Calibration & Replacement | Periodic maintenance or replacement of sensors (accelerometers, gyroscopes, etc.) used for motion tracking. | R100 |
|  |  |  |
|  |  |  |

**Total Estimated Monthly Cost:** **≈ R300**

**Two-Year Growth Projection**

The following cost estimations are based on client Sarina's feedback, keeping in mind that the solution in place was built on top of last year's project base. Thus, most of the development infrastructure, hosting, and setup fees have already been incurred. The rest of the costs are largely confined to minor maintenance, software upgrades, and small-scale hardware replacements.

Growth is forecast conservatively 5% year one and 10% year two, reflecting low operations scaling as user engagement and data storage accrue over the years.

|  |  |  |  |
| --- | --- | --- | --- |
| Year | Estimated Monthly Cost (ZAR) | Annual Cost (ZAR) | Growth Justification |
| Year 1 | R300 | R3,600 | Maintenance-only period. Small improvements, testing, and low operational expense. |
| Year 2 | R330 *(~10% increase)* | R3,960 | Moderately more expense with more use of data, slight feature improvements, and periodic hardware upgrades. |

**Change Management**

Technical success of the MoveWithMe project depends not just on technical success, but also on the successful adoption of the system by the organization and end-users. Change management within this project is designed to walk all the stakeholder's management, developers, parents, and children through adapting to use the smart wearable system and its associated mobile app.

Our approach emphasizes open communication, training, feedback integration, and long-term support such that both the organization and users fully embrace the technology and its benefits.

1. Organizational Adoption

The organization will adopt the MoveWithMe intelligent wearable system as part of a broader goal to develop child wellness, safety, and activity tracking through innovation. Adoption is warranted due to the need for current, data-oriented technology that can provide insights into children's movement patterns and overall activity.

How and Why the Organization Will Adopt

Innovation and Alignment of Research: The project aligns with the mission of the company to apply wearables in research of child development and parent support initiatives.

* Operational Effectiveness: Data collection and analysis are streamlined by the intelligent wearable system, reducing manual reporting and facilitating better tracking of health and developmental outcomes.
* Data-Informed Decision Making: Monitoring and analysis in real time allow supervisors and researchers to make evidence-based recommendations on children's activity and development.
* Ethical and Safe Deployment: The solution aligns with ethical and regulatory standards to support the organization's responsible adoption of the product.

Implementation Steps:

* Training of staff to understand device calibration, data management, and ethics.
* Pilot roll-out and phased roll-out for easy integration with current infrastructure.
* Regularly scheduled review meetings with managers and stakeholders (Sarina and Yusuf) to monitor adoption.

2. User Adoption

The users the parents, caregivers, and children will adopt the MoveWithMe system because it is usable, interactive, and of real value in helping people develop healthier habits. The wearable and mobile phone application are designed to be woven into and enhance daily lives and provide helpful feedback on activity.

How and Why Users Will Adopt

* Ease of Use: The mobile app is designed with a user-friendly UI/UX design, making it easy for parents to view and understand the movement data of their child.
* Real-Time Feedback: The wearable sends instant notification and visual feedback (e.g., LED lights and sound) via the companion app to allow children to enjoyably and interactively monitor activity.
* Health Awareness and Motivation: The system encourages children to be more active using gamified elements and reassures parents in the form of clear data visualization.
* Accessibility: The product is made to be affordable and durable, hence accessible to different socio-economic groups.

 Implementation Measures:

Step-by-step onboarding instructions and tutorial videos for initial setup to help parents and children get started.

Gathering feedback during testing stages to ensure continuous improvement in usability and comfort.

Partnering with occupational therapists and neonatologists to ensure that the product meets health and safety standards.

3. Adoption Gain Strategy (Organization + Users)

To facilitate easy adoption at both organization and user levels, the team designed a three-stage change management strategy focusing on education, involvement, and on-going improvement.

Phase 1: Education & Awareness

Conduct workshops and briefing sessions for stakeholders, supervisors, and end-users.

Provide documentation, setup guides, and FAQs for both technical and non-technical users.

Highlight benefits such as health monitoring, safety data, and ease of use.

Phase 2: Engagement & Pilot Testing

Conduct pilot testing with the assistance of schools and rural communities in the presence of supervisory guidance.

Gather qualitative feedback from parents and children to evaluate comfort, dependability, and satisfaction.

Adjust technical features and software functions based on actual world feedback.

Phase 3: Continuous Support & Improvement

Leave communication channels open (WhatsApp, Microsoft Teams, GitHub) for ongoing assistance.

Provide post-deployment monitoring for smooth performance and user satisfaction.

Encourage feedback-guided upgrades and potential feature extension via version-controlled releases.

**Data Schema Documentation**

1. Data Storage Strategy Overview

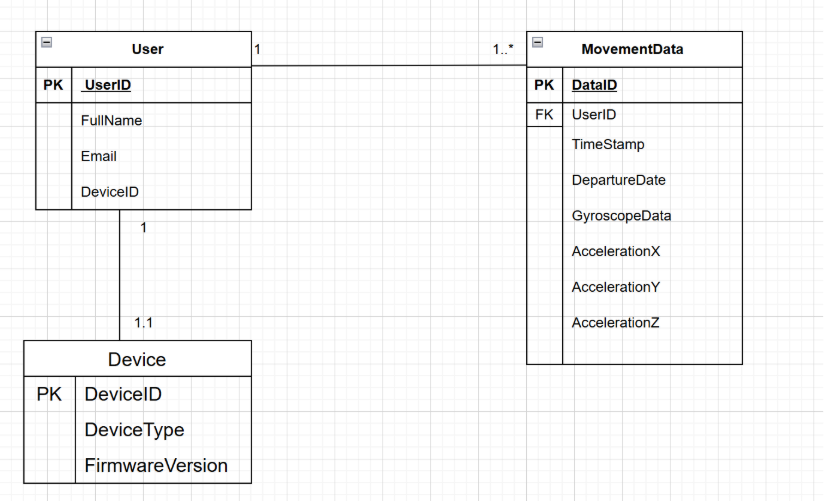
The system was designed with simplicity, scalability, and real-time synchronization in mind. As the application collects sensor data and keeps user-related information, our team chose to implement a cloud-based NoSQL database architecture. This allows the application to efficiently handle semi-structured data such as movement readings, device configurations, and user profiles without requiring a predefined relational schema.

The chosen technologies are:

1. Firebase Realtime Database / Firestore – for offering real-time data synchronization between app and cloud.
2. Local SQLite Cache – for offline data storage and syncing when the network is available.
3. JSON format – for data storage and exchange due to its lightweight, flexible nature, and Android Studio and REST API support.

This hybrid data approach offers both performance and reliability, even in low-connectivity areas.

2. Entity Relationship Diagram (ERD)



Explanation of Entities:

User: Represents a person who is wearing the smart clothing or using the app. Contains identification and registration details.

MovementData: Contains recorded sensor data (accelerometer, gyroscope) with timestamps. Linked to the user via UserID.

Device: Represents the hardware (phone or ESP32) which is recording the data, linked to the user's account via DeviceID.

This structure allows for data integrity, with space to add new sensors or metrics in future updates.

3. JSON Schema (for NoSQL Storage)

In Firebase or a similar document-based system, each record is stored as a JSON document. Example schema is shown below:



JSON Usage advantages:

Schema-less and free-flowing, allowing new sensor fields to be added with ease.

Most appropriate for integration with Android applications using libraries like Retrofit or Gson.

Lightweight and supports online (Firebase) as well as offline (SQLite) applications.

4. Data Flow Description

Data Collection:

Movement data (i.e., acceleration, rotation) is captured by the ESP32 microcontroller and mobile sensors.

Local Storage (Offline Mode):

Data is temporarily buffered in SQLite on the mobile phone for offline reliability.

Synchronization (Online Mode):

Every data is pushed into Firebase/Firestore in JSON format once the connection re-establishes.

Cloud Processing:

Stored data can be visualized, analysed, or exported for additional research work.

5. Rationale for Technology Choice

|  |  |  |
| --- | --- | --- |
| Technology | Purpose | Reason for Selection |
| Firebase | Cloud-based database | Real-time sync, scalability, and seamless integration with Android. |
| JSON | Data exchange format | Lightweight and compatible with APIs. |
| Android Studio | App development | Native integration with Java/Kotlin and Firebase SDKs. |
| SQLite | Local storage | Offline data access and minimal setup. |

**Architecture artifacts**

1. What architecture patterns were used and why

Patterns used mainly

Client–Server (Mobile + Cloud) — mobile app (Kotlin) as the client; Firebase services as the server.

Why: clean separation of concerns, centralized user management and storage, built-in scaling and real-time capabilities which fit MoveWithMe's cross-session access and supervisor monitoring requirement.

Offline-First with Sync-When-Available — local storage on Android device (Room) with background sync to cloud (Firestore or Realtime DB) when online.

Why: field testing in low-connectivity rural areas; offline-first guarantees data capture and good UX even with spotty network.

Event-Driven / Stream Processing — sensor readings as event streams; UI and cloud components react to new events (visualization, logging, analytics).

Why: motion data is time-series and requires low-latency responses (animated UI, LED feedback) and batch/cloud-efficient computing.

Layered Architecture (Presentation / Domain / Data) — UI (Kotlin MVVM) / Domain (movement-analysis, validation) / Data (Repository, Room, Firestore).

Why: maintainability, testability and clean responsibility boundaries among UI, business logic and persistence.

2. Design Patterns

MVVM (Model-View-ViewModel) — used in the Android UI (Kotlin).

Why: keeps UI isolated from business logic; interplays nicely with LiveData/Flow for sensor data in real-time and is unit-test friendly.

Repository Pattern — single abstraction of data access (local Room or cloud Firestore).

Why: isolates data source complexity from ViewModels and re-unifies sync/batching logic.

Observer / Publish-Subscribe (LiveData / Kotlin Flow) — UI components subscribe to sensor/movement event streams.

Why: enables immediate updates of UI (animated character, indicators) upon received sensor events.

Singleton — for cross-cutting services (Connectivity manager, Sensor manager, Auth manager).

Why: offers single, consistent instances for heavy-weight or globally accessed services.

Factory (or Strategy) for Sensor Handlers — factory creates handlers for accelerometer, gyroscope, etc.; strategy encapsulates processing algorithms.

Why: supports multiple types of sensors and provides easy integration of new sensors in a safe manner.

Dependency Injection (Hilt/Dagger) — inject ViewModels, services, and repositories.

Why: improves testability and decouples component creation from usage.

Adapter — to convert domain data into UI elements (charts, lists).

Why: keeps presentation code clean and reusable.

Command (light use) — encapsulate user actions or periodic sync jobs as commands where possible.

Why: allows queuing, retry and undoable operations in the sync pipeline.

3. Architecture Patterns

Chosen overall architecture:

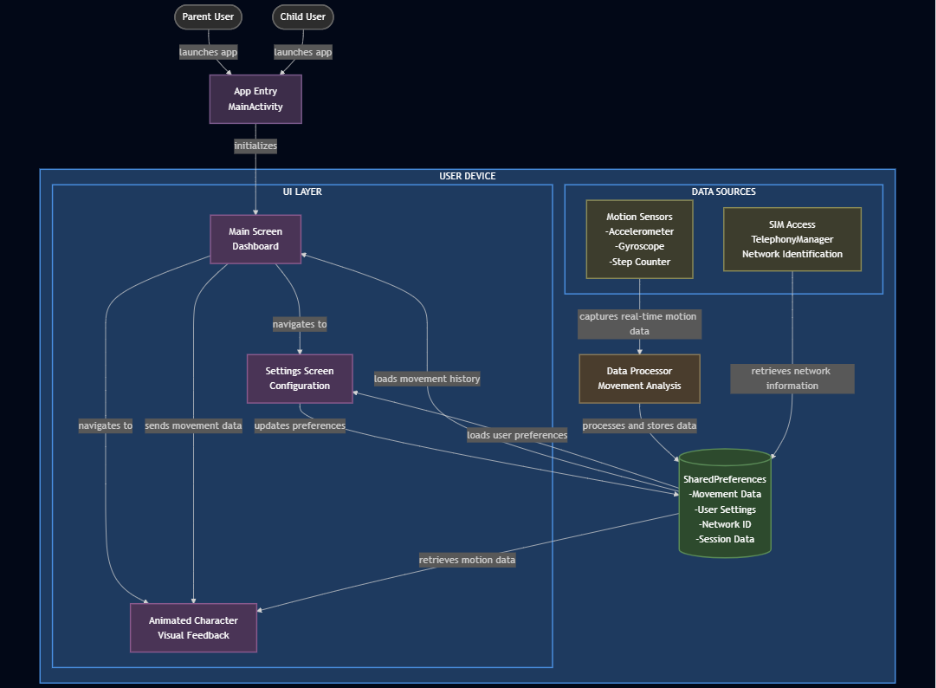
A client-server + offline-first, event-driven structure with a layered codebase on the client. Major features:

* Client responsibilities
* Sensor data capture (ESP32 or device sensors).
* Local buffering and storage (Room) for offline reliability.
* Instant UX responses (UI animation, LED feedback).
* Authentication and secure communication to cloud services.
* Edge / Connectivity responsibilities
* Connectivity module (ESP32) can stream to phone via BLE or local Wi-Fi, or (as an alternative) push to cloud when directly connected.
* Does minimal pre-processing or batching to reduce network utilization.
* Cloud responsibilities
* Authentication (Firebase Auth).
* User profiles and movement events persistent storage (Firestore or Realtime DB).
* Realtime synchronization for real-time monitoring.
* Serverless Cloud Functions for heavier processing (analytics, anomaly detection) or timed jobs.

Why this architecture

Maximizes reliability (offline-first), scalability (serverless cloud), and real-time capability (Firebase Realtime / Firestore). Reduces DevOps overhead allowing the team to focus on algorithms and testing.

Architecture Diagram



**Diagram Explanation**

**Top: Users**

**Parent User / Child User**   
Both the parent and child can launch the application independently.

* **Parent User** accesses the app to monitor and configure child movement settings.
* **Child User** launches the same app, which provides feedback and tracks their movements during activities.   
  There is **no cloud authentication** all access and data remain **local to the device** for simplicity and privacy.

**Centre: App Entry Point**

**App Entry (MainActivity)**   
This is the main entry activity that initializes the app’s UI and device connections. It sets up the motion sensors, loads local data, and directs users to the **Main Screen (Dashboard)**.

**Left: UI Layer (User Device)**

**Main Screen (Dashboard)**   
The central interface displaying live movement feedback and providing navigation options. It retrieves local session data, user preferences, and motion updates.

**Settings Screen (Configuration)**   
Allows the parent user to modify user settings, view movement history, and update local preferences. Changes are stored locally using SharedPreferences.

**Animated Character (Visual Feedback)**   
A fun, kid-friendly component that responds to the child’s motion data in real time giving immediate, visual feedback based on detected movements.

**Right: Data Sources**

**Motion Sensors**   
Includes the **accelerometer**, **gyroscope**, and **step counter**, capturing real-time motion data directly from the phone or connected ESP32 device.

**SIM Access (TelephonyManager & Network Identification)**   
Retrieves basic network or SIM-related identifiers to uniquely associate sessions with the current device (used only locally for session tracking).

**Middle-Right: Data Handling**

**Data Processor (Movement Analysis)**   
Processes raw sensor data into readable movement patterns. It stores processed results and metadata in local storage for later viewing or visual feedback.

**SharedPreferences (Local Data Store)**   
Acts as the lightweight local storage for:

* **Movement Data**
* **User Settings**
* **Network ID**
* **Session Data**
* This replaces any need for cloud-based storage, ensuring all information stays **within the device** for security and offline reliability.

4. Cloud Architecture

The app's cloud architecture was envisioned to be scalable, secure data, and real-time performance while keeping the feature of ethical and privacy compliance. Firebase was selected as the primary cloud platform due to its intrinsic ecosystem, simplicity in integrating it with Android Studio, and its support for authentication, real-time data synchronization, and secure storage of data.

Cloud Services and Technology Decisions

The following cloud services were selected to add the system's backend capabilities and data storage:

|  |  |  |
| --- | --- | --- |
| Service | Purpose | Technology Decision Rationale |
|  |  |  |
| Firestore Database | Primary document-oriented database storing user data, device metadata, and motion events. | Offers real-time updates, scalable data storage, and advanced query capabilities, making it ideal for structured mobile data. |
| Realtime Sync (Firebase SDK) | Provides low-latency synchronization between parent and child devices and supervisor dashboards. | Uses WebSockets/gRPC to maintain instant updates across clients. |
| Cloud Functions | Executes server-side logic, such as data aggregation, notifications, and validation. | Enables backend automation without dedicated server infrastructure. |
| Cloud Storage (Optional) | Stores large files such as anonymized logs or exported reports. | Supports secure file uploads and integrates with Firebase Auth for access control. |

**Functional Requirements**

The following are the functional requirements that identify the major capabilities and behaviours that the Child Movement Tracking System ought to provide. These requirements ensure that the system is operational in meeting the client's needs for functionality, usability, and reliability.

1. Child Movement Tracking Using Sensors

The system ought to track and monitor the physical movement of a child through the incorporation of in-built motion sensors.

The sensors will collect movement data constantly in real-time.

The program will process and display this data in an understandable manner.

The tracking functionality will function both indoors and outdoors and deliver smooth performance under either condition.

2. Animated Visual Feedback Based on Movement

The program will provide animated visual feedback proportional to the child's rates of detected movement.

As the child moves around, the interface will display animated markers or graphics that physically represent the degree of activity.

The visual feedback will dynamically change according to movement changes, maximizing user interest and understanding.

This alternative is to make the app more interactive and informative to parents and guardians.

3. Switch Between Online and Offline Modes

The system will be integrated with both online and offline working modes in order to offer uninterrupted functionality.

In online mode, data will be uploaded and synced with the cloud (Firebase).

Offline mode will cache information locally on the device and sync it whenever connectivity resumes.

The user will be offered the option to manually or automatically switch between these modes based on the network availability.

4. SIM Card Support

The system will support SIM cards to enable identification and fundamental network-related functionalities.

The application will recognize and utilize the basic identification or network information of the SIM card.

This feature will assist user authentication and location-based service when required.

The SIM capability will never take nor transmit any sensitive private data, always complying with privacy legislations.

5. Local Data Storage with Syncing Support

Local data storage will be provided in the app to retain tracking information even when the internet is not available.

Data obtained during offline usage will be stored securely on the device.

When reconnecting, the system will automatically sync local data with the online database.

The syncing process will be such that no data duplication or loss is encountered during the transfer.

**Appendices**

Declaration of Authenticity



Ananta Reddy (Project Manager)

ST10143151



Aiden Reddy (Backend Developer)

ST10290935



Shreya Dhawrajh (Project Administrator)

ST10249644



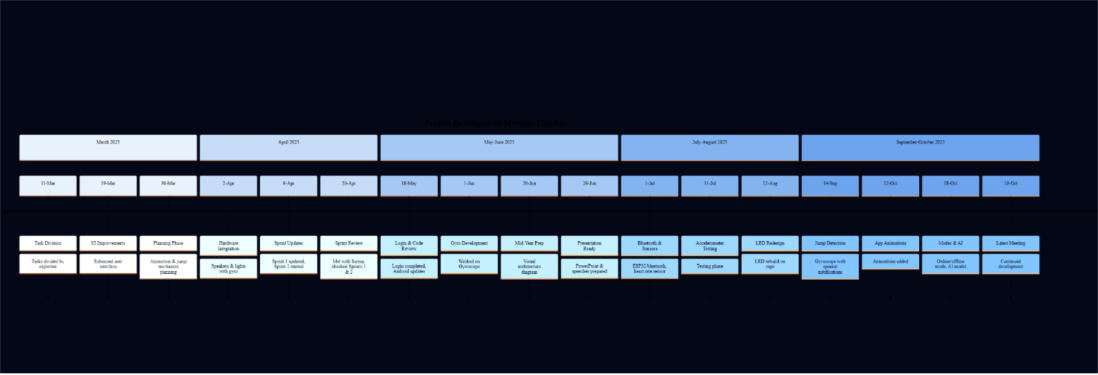
Dheyan Rambali (UI/UX Designer)

ST10248202

Matthew Mason (Full Stack Developer)

ST10403514

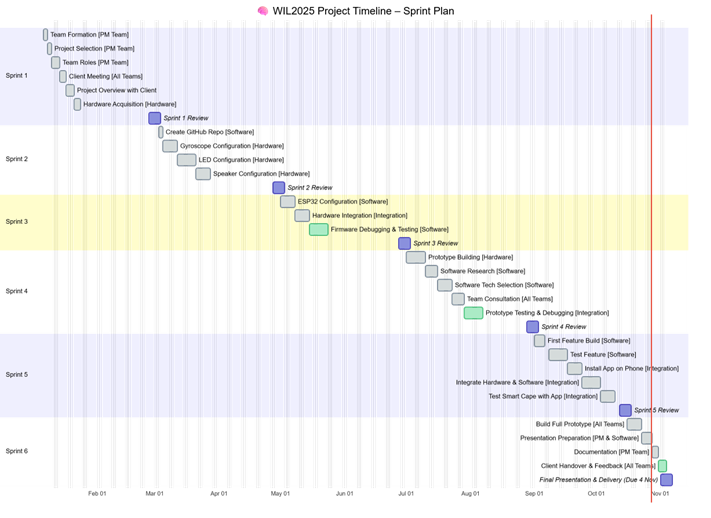
Project Timeline



[**https://1drv.ms/w/c/4b361af379caf760/EbzcAZqZqr9LglvpMR2xpUcBo6QUyfwWza5I283Q3jkRmQ?e=qsA7wc**](https://1drv.ms/w/c/4b361af379caf760/EbzcAZqZqr9LglvpMR2xpUcBo6QUyfwWza5I283Q3jkRmQ?e=qsA7wc)**- Sprint Doc**

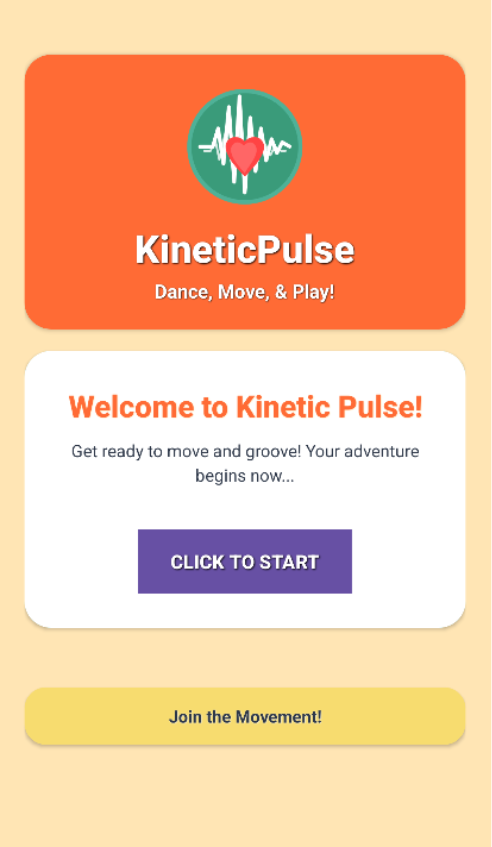
[WILL MEETING MINUTES.xlsx](https://1drv.ms/x/c/4b361af379caf760/EQ0znbOir0NIqtKfn9FPB4kBI1s0q9NEBDkFpSji6WW3rQ?e=vcpsf8&nav=MTVfezRFMjE4RkIzLUI3MUMtNEU1Qi05OTk3LTY3RDNFRjc2MDFERn0)   **- Meeting Minutes Doc**

 Gantt Chart



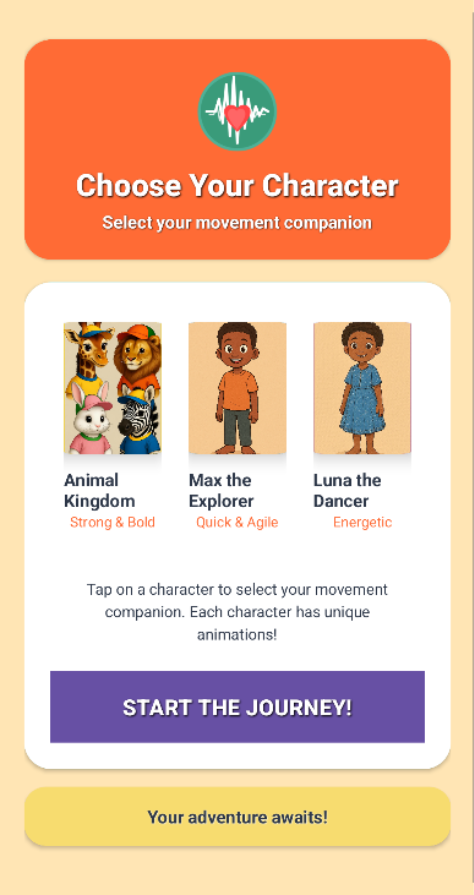
**User Experience Journey**

Welcome Screen



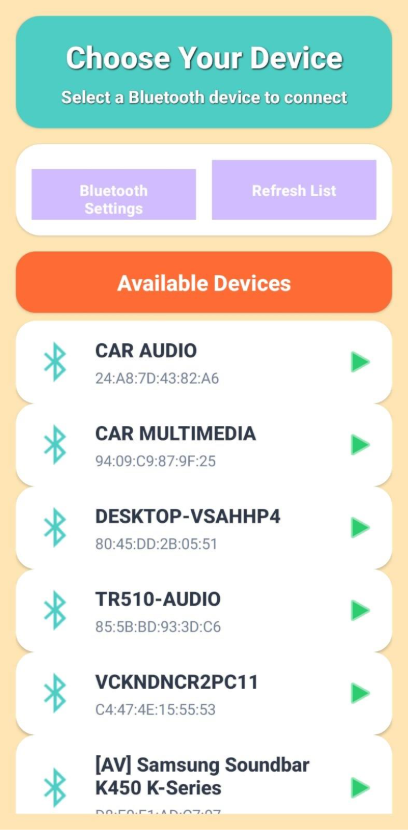
When launching the application, this is the screen that the user will be greeted with. It features the application logo and warm/vibrant colours for the UI elements. In the centre, is the ‘Click to Start’ button. This button will navigate the user to the ‘Character Select’ screen.

Character Selection Screen



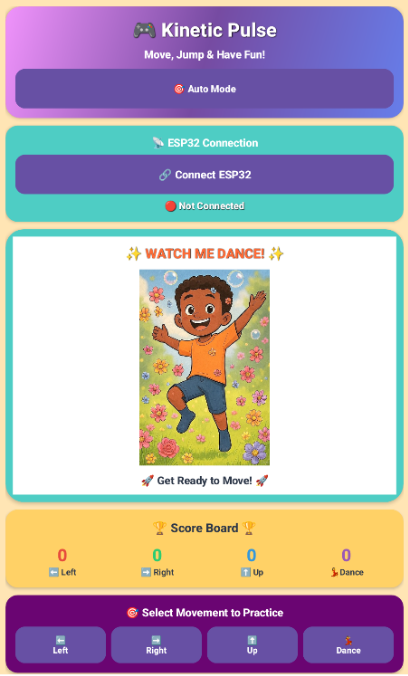
This is the screen that the user will see after tapping the ‘Click to Start’ button on the ‘Welcome’ Screen. The user is shown 3 options that displays characters that they can tap to select from. These characters will be displayed on the ‘Terminal Fragment’ screen. The large button that says ‘Start the Journey’ will navigate the user to the ‘Bluetooth Devices’ screen when tapped.

Bluetooth Devices Screen



 This is the screen that the user will see after tapping the ‘Start the Journey’ button on the ‘Character Select’ screen. This screen will display a list of all the available Bluetooth devices when Bluetooth is on. If Bluetooth is off, a pop-up will appear prompting the user to enable Bluetooth. The pop-up will have the options ‘Not Now’ and ‘Enable.’ If ‘Not Now’ is selected, the pop-up will disappear, and the app will display a message in the section that the device list is, saying that “Bluetooth is disabled. Tap ‘Bluetooth Settings’ to enable.” Tapping the ‘Bluetooth Settings’ button will bring up a pop-up on your phone that says that the app is asking to turn on Bluetooth (This pop-up will also appear after selecting ‘Enable’ on the applications own pop-up). You can deny or allow this. After allowing it to turn on Bluetooth, the list should refresh and show the list of available Bluetooth devices. After tapping on an available device, the user will be taken to the ‘Terminal Fragment’ screen.

Terminal Fragment Screen

This is the terminal fragment screen. This screen is what the user will see after selecting a Bluetooth device on the ‘Bluetooth Devices’ screen. At the top of the screen, is a button that changes the movement detection mode when tapped. It has ‘Auto Detect’ for picking up all movement, ‘Phone Gyro’ for picking up movements using the users phone gyroscope, and it has ‘ESP32’, which will use the connected ESP32 device to detect movements. Below this, is the button for connecting to the ESP32. The status of the connection is also shown here.

In the centre of the screen is the area where the chosen character and their animations are played. This takes up a large portion of the screen as it is the main visually engaging feature of the application. Below this, is the scoreboard that tracks the number of each movement done. Each movement counter (number) is a different colour. Below the scoreboard, is the buttons for selecting a movement. When a user selects a movement by tapping on one of the buttons (e.g. Left), the corresponding animation for that movement will play. After a successful movement, a toast message saying “Perfect! Movement Completed!” will be displayed and the scoreboard will be updated for the corresponding movement done. There are also milestone badges for the user reaching a certain number of movements. These will be displayed on the scoreboard.

The second image shown is part of the screen when scrolling down on the terminal fragment screen. When scrolling down, below the movement buttons, is the song/music player. The user can tap on the area with the song name, which will bring up a list of songs that is included within the application. The user can then tap on their desired song. The user can tap the ‘Play Song’ button below it to play the selected song. They can then tap the ‘Stop’ button to stop the song that is currently playing. If the user wants to play their own music through their desired music app, they can tap the ‘Open Music App’ button. This will display a pop-up that shows the available music player apps on their phone. They may tap on the desired app to launch it and play music.

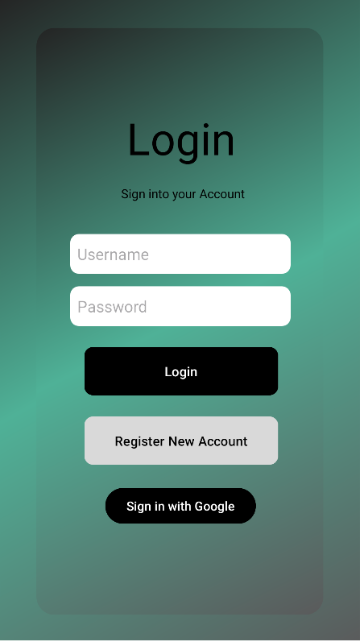
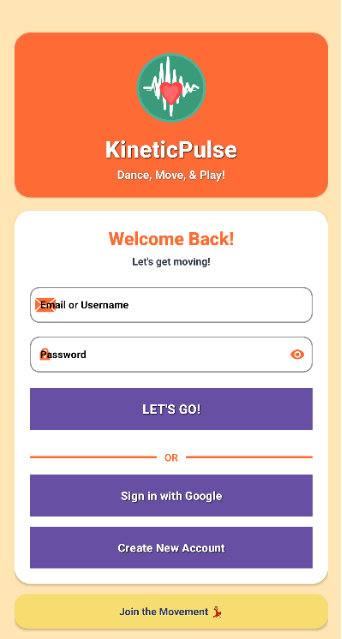
Below the music section, is the LED controls. This panel includes multiple buttons that allows the user to change the colours of the LED of the connected cape device.

Below this, is the ‘AI Movement Classifier’ section. After performing a movement, this section will update with a message of what type of movement the AI detected, while also displaying its level of confidence in the movement is classified.

**Comparing Original UI with the Updated UI**

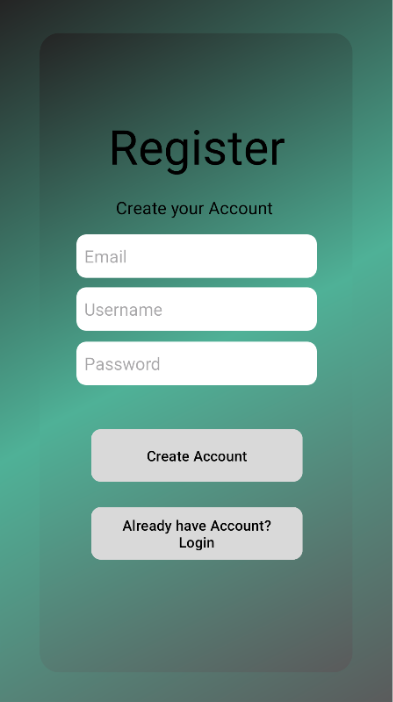
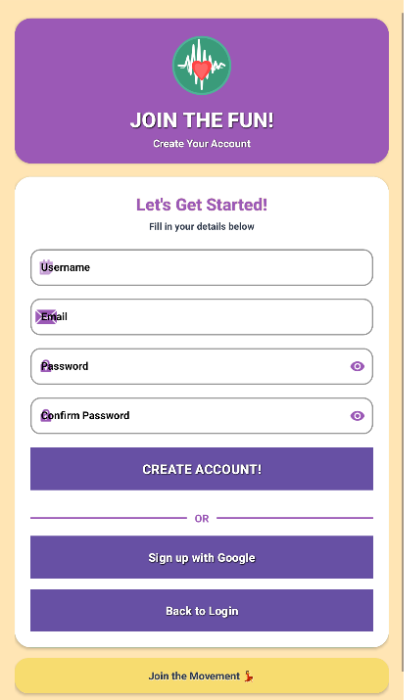
**Note**: Original UI is the picture on the left, with the updated Ui being the picture on the right.

Login Screen

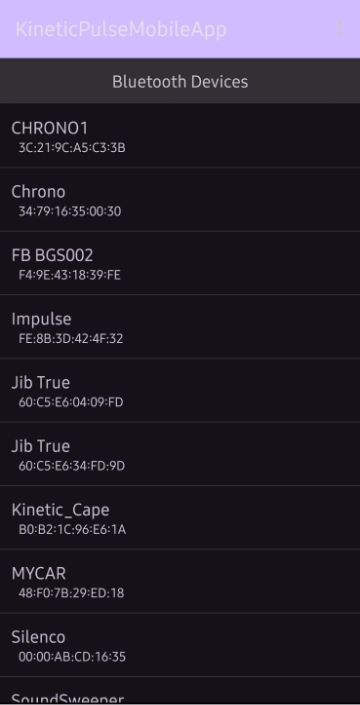
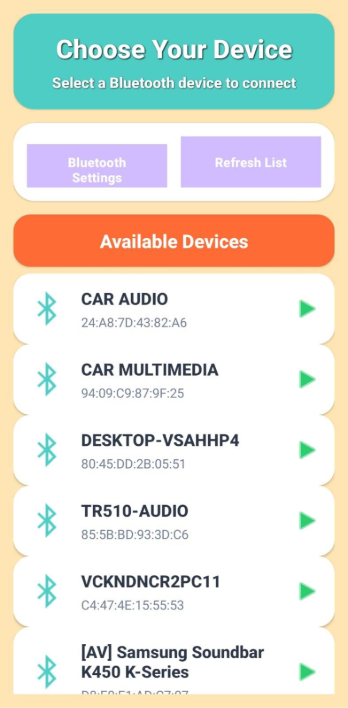
The updated Login screen replaces the old ConstraintLayout with a scrollable ScrollView containing a ConstraintLayout and multiple CardViews. The header now includes a logo, app title, and subtitle, giving the screen more branding and visual appeal. The input fields are upgraded to Material TextInputLayouts with icons, rounded corners, and a password toggle, making them more modern and user-friendly. The buttons are full-width, elevated, and use a colour that stands out on the updated screen. A divider separates login options, and a footer card adds a fun motivational message. Overall, the new layout is more vibrant, organized, and visually engaging while keeping all the original functionality.

Register Screens

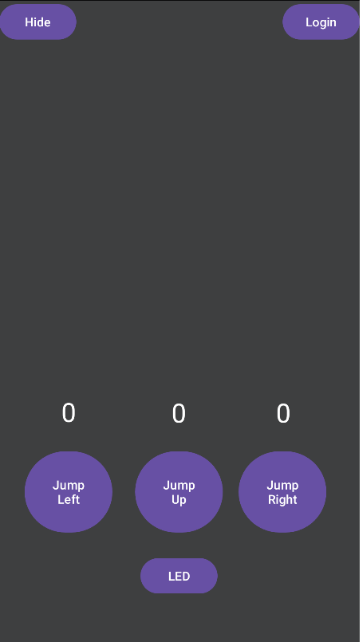
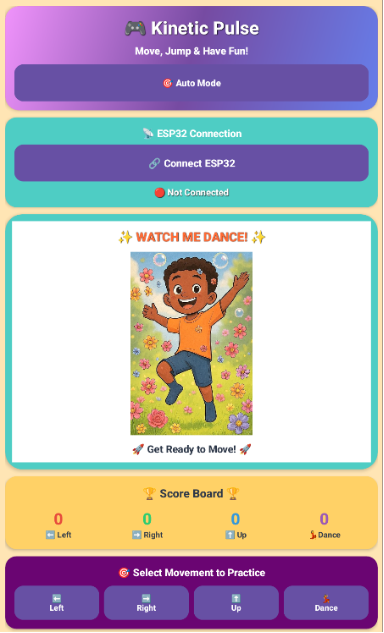
The original Register screen is a simple ConstraintLayout with three plain EditText fields for Username, Email, and Password, two buttons for creating an account and going back to login, and minimal styling. Labels like “Register” and “Create your Account” are fixed TextView’s with basic font sizes and colours. The updated Register screen adopts the same modern card-based style as the updated Login screen. It wraps everything in a ScrollView for better scrolling on small devices, adds a colourful header card with a logo and title text, and uses a main registration card containing TextInputLayout fields for Username, Email, Password, and a new Confirm Password, all styled with rounded corners, stroke colours, icons, and password toggles. Buttons for registration, Google Sign-In, and returning to login are now visually consistent with shadows, bold fonts, and coloured backgrounds, while a divider labelled “OR” separates the options. A footer card remains at the bottom with the message “Join the Movement,” matching the updated aesthetic. Overall, the update modernizes the layout, improves input styling, adds a confirm-password field, and creates a cohesive, card-based visual hierarchy consistent with the Login screen.

Device List Screen

The updated Device List Item layout was redesigned to enhance visual appeal and create a more vibrant, child-friendly interface. The original design used a simple vertical LinearLayout containing two TextViews, which resulted in a plain and minimal appearance. In the revised version, a CardView was introduced as the root element to provide a more structured and modern layout, featuring rounded corners and elevation for visual depth. This change helps each device entry stand out clearly as an individual, interactive component. Additional visual elements, such as a Bluetooth icon and a navigation arrow, were included to guide users intuitively and make the interface more engaging. The use of bright colours, including teal and green, contributes to a lively and playful atmosphere that is more appealing to children. Text styling was improved by using bold and larger font sizes for device names and softer tones for secondary text, enhancing readability and visual hierarchy. Consistent padding, spacing, and alignment further improve overall presentation and usability. Collectively, these adjustments result in a user interface that is more dynamic, approachable, and suited to a younger audience.

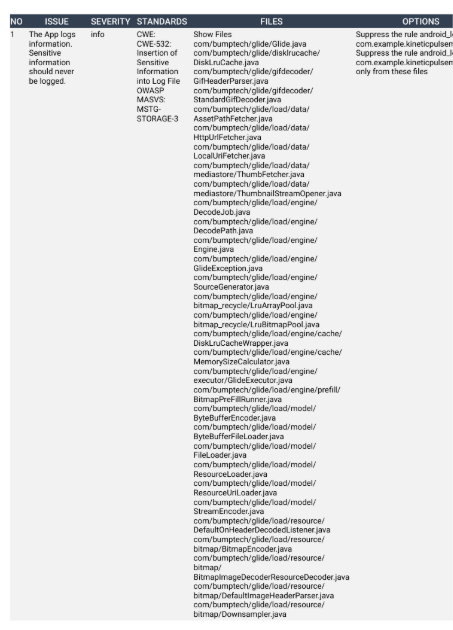
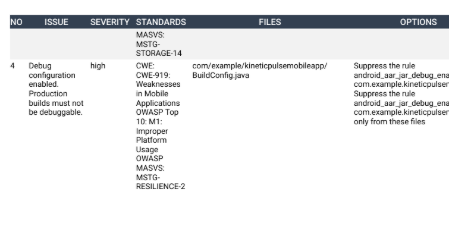
Terminal Fragment Screen

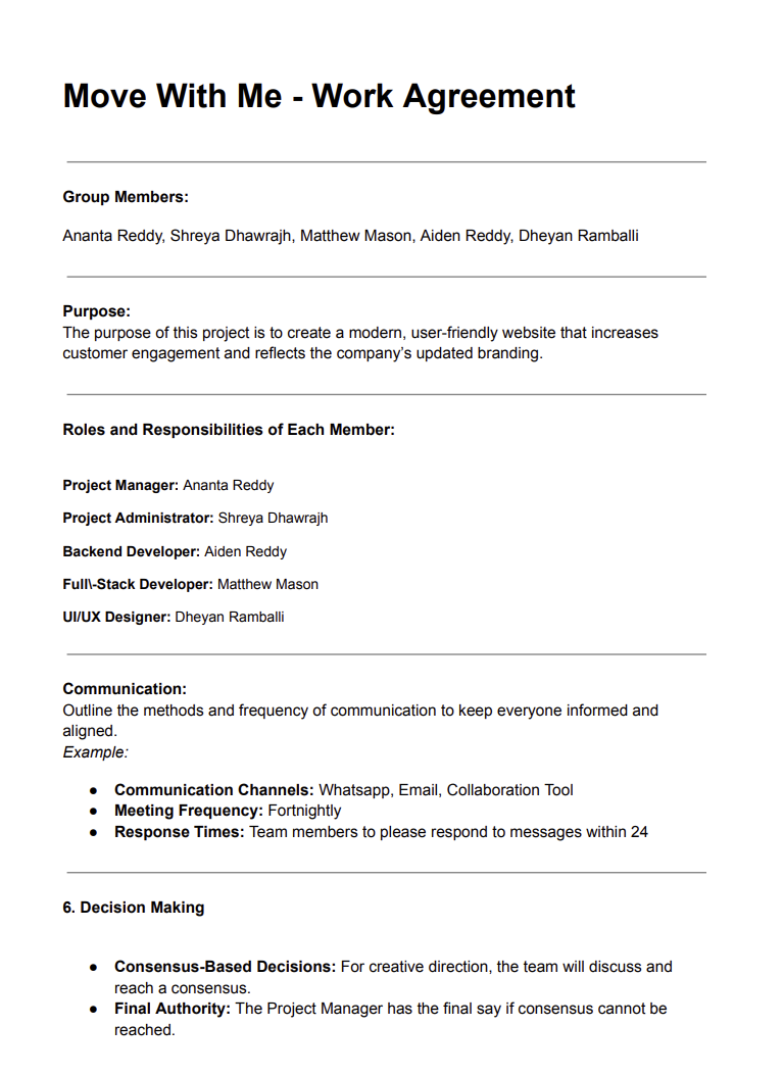
The original Terminal Fragment screen layout was a functional but basic interface, primarily structured as a vertical LinearLayout with a hidden ConstraintLayout for buttons, a text area for incoming messages, and simple controls for sending text and toggling LED features. While it provided the necessary functionality for device interaction, movement tracking, and LED control, its appearance was plain, with minimal visual hierarchy or styling. In contrast, the updated layout transforms the interface into a visually engaging, child-friendly environment using a ScrollView to ensure accessibility on smaller screens. The new design incorporates vibrant colours, rounded CardView sections, and thematic icons to improve usability and appeal. Key features are presented in distinct, well-organized cards, including a playful header, ESP32 connection controls, character display with a placeholder or video, a detailed scoreboard, movement selection buttons, LED controls, built-in kids’ songs, and an AI movement classifier section. Each interactive element now uses modern button styles, clear labelling, and visual feedback, making the interface both intuitive and enjoyable. Overall, the redesign emphasizes engagement, clarity, and playful aesthetics while retaining all original functional capabilities.

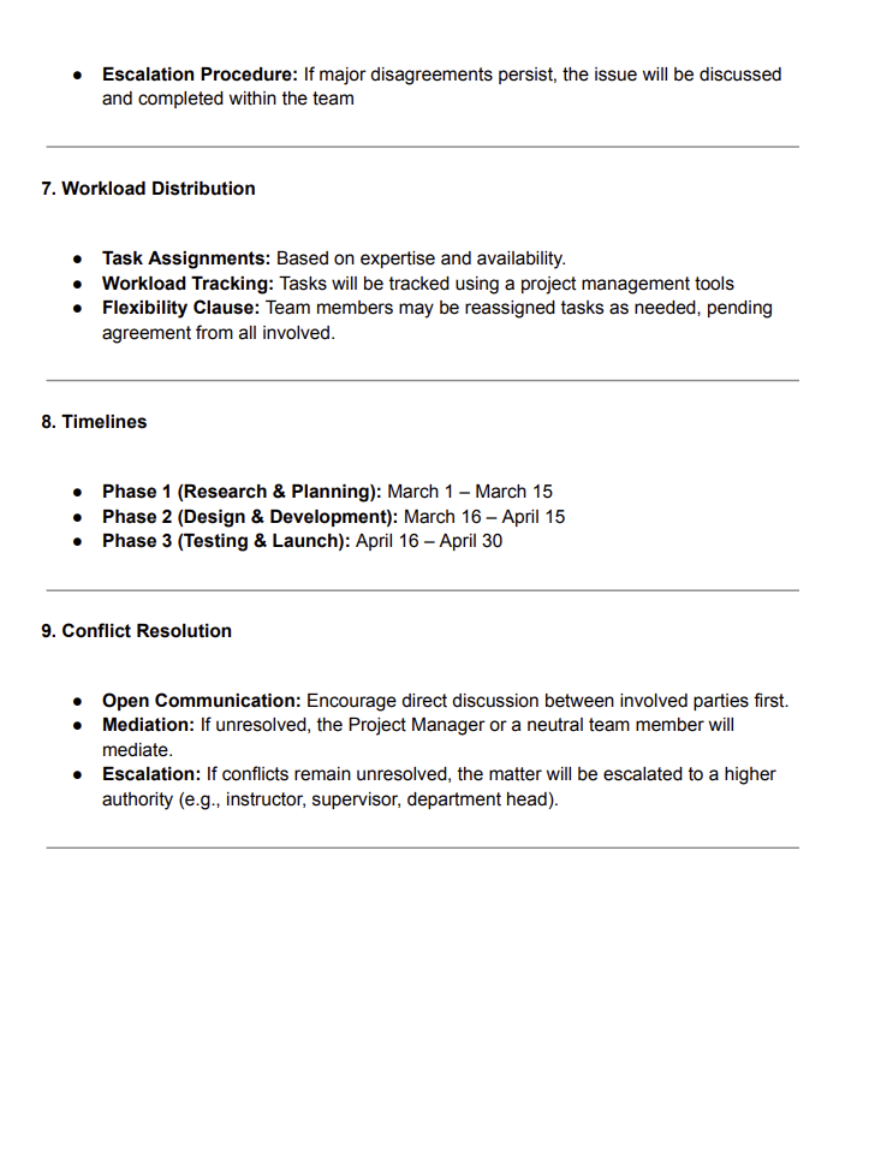
MOBSF Report

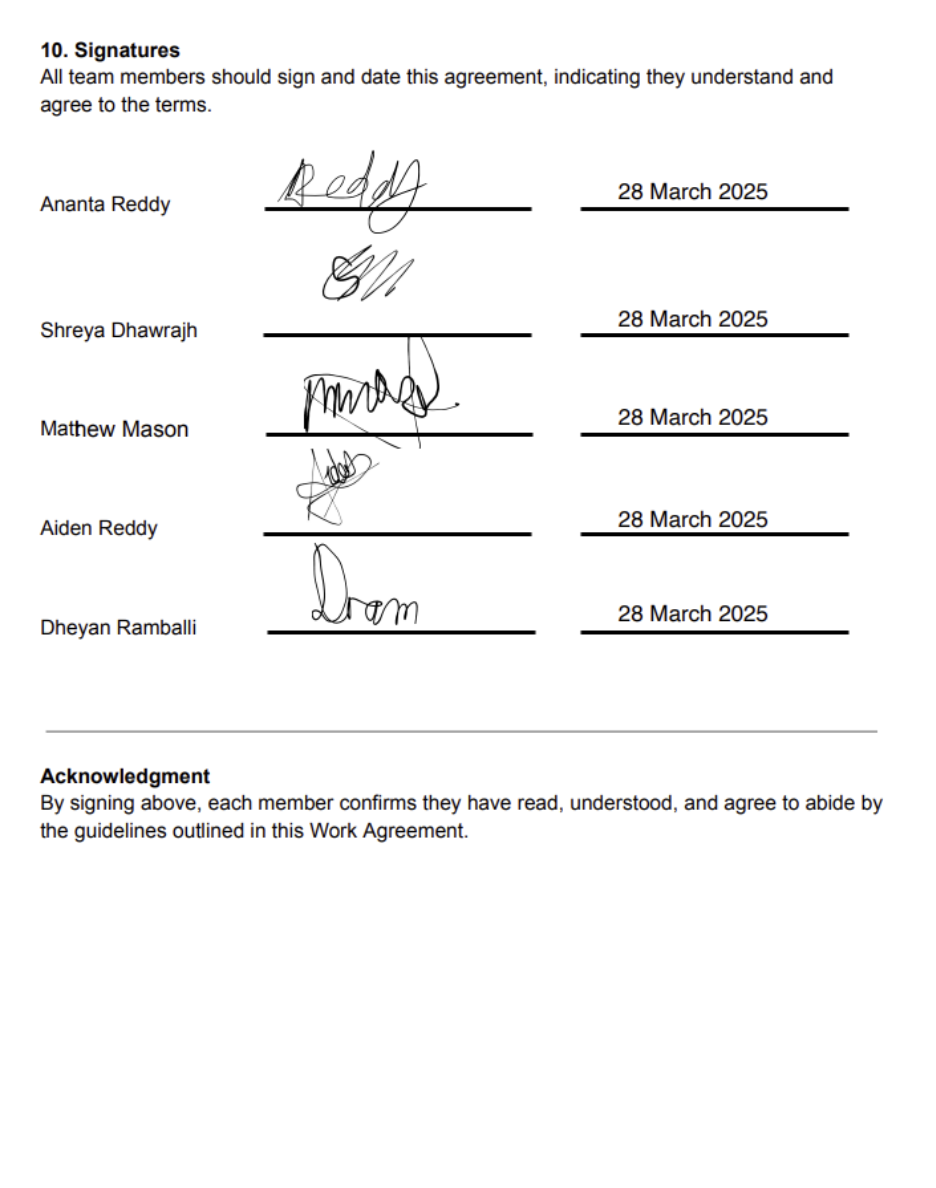




**WILL Work Agreement**







**Reference List**

 Andreas Spiess, 2017. [#173](https://www.youtube.com/hashtag/173) ESP32 Bluetooth BLE with Arduino IDE (Tutorial) and Polar H7. [video online] Available at : < <https://youtu.be/2mePPqiocUE?si=Up7wtPTHQwVUEOI8>> [Accessed 12 September 2025].

Android Developers, 2025. Add menus. [online] Available at: <<https://developer.android.com/develop/ui/views/components/menus#fragments>> [Accessed 12 July 2025].

Android Developers, 2025. Asynchronous work with Java threads. [online] Available at: <<https://developer.android.com/develop/background-work/background-tasks/asynchronous/java-threads>> [Accessed 02 September 2025].

Android Developers, 2025. Behavior changes: Apps targeting Android 12. [online] Available at: <<https://developer.android.com/about/versions/12/behavior-changes-12>> [Accessed 20 July 2025].

Android Developers, 2025. Bluetooth permissions. [online] Available at: <<https://developer.android.com/develop/connectivity/bluetooth/bt-permissions>> [Accessed 12 September 2025].

Android Developers, 2025. Color. [online] Available at: <<https://developer.android.com/reference/android/graphics/Color#parseColor(java.lang.String)>> [Accessed 20 April 2025].

Android Developers, 2025. Enum. [online] Available at: <<https://developer.android.com/reference/java/lang/Enum>> [Accessed 09 August 2025].

Android Developers, 2025. Localize your app. [online] Available at: <<https://developer.android.com/guide/topics/resources/localization>> [Accessed 12 July 2025].

Android Developers, 2025. Motion sensors. [online] Available at: <<https://developer.android.com/develop/sensors-and-location/sensors/sensors_motion>> [Accessed 15 August 2025].

Android Developers, 2025. Request runtime permissions. [online] Available at: <<https://developer.android.com/training/permissions/requesting>> [Accessed 09 August 2025].

Android Developers, 2025. SensorManager. [online] Available at: <<https://developer.android.com/reference/android/hardware/SensorManager>> [Accessed 15 August 2025].

Android Developers, 2025. Sensors Overview. [online] Available at: <<https://developer.android.com/develop/sensors-and-location/sensors/sensors_overview>> [Accessed 18 August 2025].

Android Developers, 2025. SpannableStringBuilder. [online] Available at: <<https://developer.android.com/reference/android/text/SpannableStringBuilder>> [Accessed 02 September 2025].

Android knowledge, 2023. Fragment in Android Studio using Kotlin | Android Knowledge. [video online] Available at: < <https://youtu.be/fT4MVQZWPRg?si=q2bBwA3vmjdSwaNN>> [Accessed 12 July 2025].

Arduino Titan, 2024. MPU6050 Gyroscope+OLED display Interfacing with ESP32 | Wokwi Esp32 simulator | Arduino Titan. [video online] Available at: < <https://youtu.be/KRoAsYJScEc?si=WdXZIPNDkbzj0guN>> [Accessed 09 August 2025].

ATECHS, 2025. MPU6050 Explained | Motion Detection with ESP32 and Arduino!?. [video online] Available at: < <https://youtu.be/rpCcZZJeodY?si=Ka3BX6-anw7aJjSY>> [Accessed 09 August 2025].

BINARYUPDATES, 2023. How to Setup and Program ESP32 Microcontroller– Complete Guide. [video online] Available at: < <https://youtu.be/AitCKcyjHuQ?si=kBwSdBYFv9w4CAeU>> [Accessed 12 September 2025].

Bharath kumar, 2023. ESP-32 with MPU-6050 Accelerometer, Gyroscope and Temperature Sensor (Arduino). [video online] Available at: < <https://youtu.be/1Su4HbPHZn0?si=p3EW1X_HQyGMlSwu>> [Accessed 12 September 2025].

Brand Store, 2024. How to Get google-services.json File from Firebase for Your App (Step-by-Step Guide). [video online] Available at: < <https://youtu.be/w1EPb-Jl1_U?si=wKFNSWMm6zv4f2sk>> [Accessed 15 April 2025].

C# Corner, 2022. How to use weightSum and layout\_weight In Linear Layout. [online] Available at: <<https://www.c-sharpcorner.com/article/how-to-use-weightsum-and-layoutweight-in-linear-layout/>> [Accessed 12 July 2025].

CapTech, 2020. Android Activity Result Contracts. [online] Available at: <<https://www.captechconsulting.com/technical/android-activity-result-contracts>> [Accessed 15 July 2025].

Chris Maher, 2022. How To Install WLED on an ESP32 Board and Connect / Control Addressable LEDs. [video online] Available at: < <https://youtu.be/TOEnFKLm9Sw?si=P9IigSvD8HCnyAGy>> [Accessed 12 September 2025].

CodePath Cliffnotes, 2015. Listening to Sensors using SensorManager. [online] Available at: <<https://guides.codepath.com/android/Listening-to-Sensors-using-SensorManager>> [Accessed 18 August 2025].

Core Electronics, 2022. How to Easily Control Addressable LEDs with an ESP32 or ESP8266 | WLED Project. [video online] Available at: < <https://youtu.be/GYxctjukehY?si=xNlYUkqMdVG-c4uX>> [Accessed 13 September 2025].

CS CORNER Sunita Rai, 2025. How to Connect Mobile Phone with Android Studio to Run App [2025] |Connect Phone With Android Studio. [video online] Available at: < <https://youtu.be/9Tt8uLUgSj8?si=y_yUgMfCP_UcPWGN>> [Accessed 20 March 2025].

Desktop Make, 2022. #2 How to connect ESP32 to ESP32 by Bluetooth (ESP32 Programming). [video online] Available at: < <https://youtu.be/3SjGH0J7KPY?si=3tfhtDyAHBYSE6Z2>> [Accessed 13 September March 2025].

DigitalOcean, 2022. Retrofit Android Example Tutorial. [online] Available at: <<https://www.digitalocean.com/community/tutorials/retrofit-android-example-tutorial>> [Accessed 20 July 2025].

DIY TechRush, 2022. ESP32 Bluetooth Classic - ESP32 Beginner's Guide. [video online] Available at: < <https://youtu.be/EWxM8Ixnrqo?si=SnWwI6mTLC3qOQze>> [Accessed 12 September 2025].

DEV Community, 2025. Serial communication with Android device. [online] Available at: <<https://dev.to/bleuiot/serial-communication-with-android-device-481o>> [Accessed 10 July 2025].

DroneBot Workshop, 2024. Bluetooth Classic & BLE with ESP32. [video online] Available at: < <https://youtu.be/0Q_4q1zU6Zc?si=eCU72Qw5J1fP4eLG>> [Accessed 13 September 2025].

Elconics, 2023. Send Sensor data to Firebase using ESP32 | Cloud setup | Arduino Coding | Firebase Realtime Database. [video online] Available at: < <https://youtu.be/cm-Qe2HMJGk?si=abRgEFNIQMgLDNc1>> [Accessed 13 September 2025].

Foxandroid, 2023. Fragments Implementation using Kotlin || Fragments using Kotlin || Android Studio Tutorial || 2023. [video online] Available at: < <https://youtu.be/h-NcxT697Nk?si=eMUYyDbMERJUqhAW>> [Accessed 12 July 2025].

GeeksforGeeks, 2025. Foreground Service in Android. [online] Available at: <<https://www.geeksforgeeks.org/android/foreground-service-in-android/>> [Accessed 10 July 2025].

GeeksforGeeks, 2025. How to Build an Application to Test Motion Sensors in Android?. [online] Available at: <<https://www.geeksforgeeks.org/android/how-to-build-an-application-to-test-motion-sensors-in-android/>> [Accessed 15 August 2025].

GeeksforGeeks, 2025. How to Display the List of Sensors Present in an Android Device Programmatically?. [online] Available at: <<https://www.geeksforgeeks.org/android/how-to-display-the-list-of-sensors-present-in-an-android-device-programmatically/>> [Accessed 15 August 2025].

GeeksforGeeks, 2022. How to Create an Alert Dialog Box in Android?. [online] Available at: <<https://www.geeksforgeeks.org/android/how-to-create-an-alert-dialog-box-in-android/>> [Accessed 10 May 2025].

GeeksforGeeks, 2025. How to Call a Method After a Delay in Android?. [online] Available at: <<https://www.geeksforgeeks.org/kotlin/how-to-call-a-method-after-a-delay-in-android/>> [Accessed 04 September 2025].

Github, 2025. Glide. [online] Available at: <<https://github.com/bumptech/glide>> [Accessed 02 October 2025].

How Do You? DIY, 2020. Desoldering | Soldering Basics | Soldering for Beginners. [video online] Available at: < <https://youtu.be/bG7yW9FigJA?si=PA6MNCmEAtXlbG6S>> [Accessed 02 August 2025].

InfyOm Technologies, 2021. How to Change App Language in Android Programmatically?. [online] Available at: <<https://infyom.com/blog/how-to-change-app-language-in-android-programmatically/>> [Accessed 02 September 2025].

KB CODER, 2021. Play Audio in Android Studio | Kotlin | Android Tutorials. [video online] Available at: < <https://youtu.be/gXbrjvUaHYc?si=BYq0DLBNXu_fAjpU>> [Accessed 20 September 2025].

LeftyMaker, 2017. Breadboard tutorial: How to use a breadboard (for beginners). [video online] Available at: < <https://youtu.be/W6mixXsn-Vc?si=6BlaoWsYZWSnxQSw>> [Accessed 02 August 2025].

MAINFRAME, 2021. Setting up an ESP32 with Arduino IDE. [video online] Available at: < <https://youtu.be/CD8VJl27n94?si=vpaU8xNPWKMLQYdY>> [Accessed 09 August 2025].

Mammothlnteractive, 2021. Build Android App with TensorFlow Lite Machine Learning Model. [video online] Available at: < <https://youtu.be/o5c2BLrxNyA?si=6FQGl8tCBWBbOB1B>> [Accessed 15 September 2025].

Medium, 2021. Android Runtime Permissions using registerForActivityResult. [online] Available at: <<https://medium.com/codex/android-runtime-permissions-using-registerforactivityresult-68c4eb3c0b61>> [Accessed 09 August 2025].

Medium, 2024. Establishing Connection with Bluetooth Classic in Android. [online] Available at: <<https://blog.stackademic.com/establishing-connection-with-bluetooth-classic-in-android-1f9f17c9a452>> [Accessed 12 September 2025].

Medium, 2023. Image Loading with Jetpack Compose; Glide-Coil Libraries. [online] Available at: <<https://medium.com/@selinihtiyar/image-loading-with-jetpack-compose-glide-coil-359c8d05c944>> [Accessed 02 October 2025].

Medium, 2024. Understanding Handlers, Loopers, and Message Queues in Android: A Practical Guide with Code Examples. [online] Available at: <<https://medium.com/@mohamed.ma872/understanding-handlers-loopers-and-message-queues-in-android-a-practical-guide-with-code-515162d1d792>> [Accessed 05 September 2025].

Microsoft Learn, [*s.a*.]. BluetoothDevice.Name Property. [online] Available at: <<https://learn.microsoft.com/en-us/dotnet/api/android.bluetooth.bluetoothdevice.name?view=net-android-35.0>> [Accessed 12 September 2025].

Nelson Darwin Pak Tech, 2023. simulation of esp32 with mpu6050 | simulation of esp32 with gyroscope and accelerometer. [video online] Available at: < <https://youtu.be/v5a7aZJiI-k?si=hFn18iTR5Piq7eXB>> [Accessed 09 August 2025].

Oracle, [*s.a*.]. String (Java Platform SE 8 ) - Oracle. [online] Available at: <<https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#String-byte:A->> [Accessed 02 September 2025].

Oracle, [*s.a*.]. Float (Java Platform SE 8 ) - Oracle. [online] Available at: <<https://docs.oracle.com/javase/8/docs/api/java/lang/Float.html#parseFloat-java.lang.String->> [Accessed 02 September 2025].

Programming Concepts, 2025. ESP32 IoT Project using Firebase, Android Studio and Arduino IDE | Programming Concepts. [video online] Available at: < <https://youtu.be/Z4UilZuN1cU?si=sOYQfro5EXLI8BZE>> [Accessed 15 April 2025].

Rui Santos, 2019. ESP32 Bluetooth Classic with Arduino IDE - Getting Started. [video online] Available at: < <https://youtu.be/RStncO3zb8g?si=pOiaO_HdAWeglaDn>> [Accessed 05 September 2025].

Robocraze, 2019. GETTING STARTED WITH ESP-NOW. [video online] Available at: < <https://youtu.be/B6NZWLKGoCA?si=FMnKa7NBqculLYwP>> [Accessed 15 April 2025].

Rui Santos, 2018. Getting Started with ESP32 Bluetooth Low Energy (BLE) on Arduino IDE. [video online] Available at: < <https://youtu.be/wkO-ytWVvC0?si=C8SPcKPqYNiYX_Uw>> [Accessed 25 August 2025].

Ralph S Bacon, 2022. [#239](https://www.youtube.com/hashtag/239) Using An ESP32 with 🟢🔴🔵NeoPixels - What Went Wrong? Simple Fix!. [video online] Available at: < <https://youtu.be/SStRG-_1wXc?si=Zdai79N_7MJUJsfJ>> [Accessed 13 September 2025].

Ratchets and Wrenches, 2015. How to Use a Multimeter for Beginners - How to Measure Voltage, Resistance, Continuity and Amps. [video online] Available at: < <https://youtu.be/TdUK6RPdIrA?si=KrW1cMTpsUOqKYtq>> [Accessed 22 March 2025].

Stack Overflow, 2025. How do we use runOnUiThread in Android?. [online] Available at: <<https://stackoverflow.com/questions/11140285/how-do-we-use-runonuithread-in-android>> [Accessed 09 August 2025].

Superb Tech, 2021. MPU6050 Sensor Arduino Tutorial. [video online] Available at: < <https://youtu.be/a37xWuNJsQI?si=Lc78AH2PCpMzIIBb>> [Accessed 20 June 2025].

SriTu Hobby, 2022. What is the ESP32 board and how to set up it with Arduino IDE?. [video online] Available at: < <https://youtu.be/wsKTmlipQOE?si=5WpvQltEB_nNlixh>> [Accessed 09 August 2025].

Taqnix, 2025. How to Generate Google Service JSON File for Firebase – Step-by-Step Guide!. [video online] Available at: < <https://youtu.be/ixp7X9eaVFc?si=KM_1eRwNF01LbzBP>> [Accessed 15 April 2025].

TensorFlow, 2018. Add TensorFlow Lite to your Android App (TensorFlow Tip of the Week). [video online] Available at: < <https://youtu.be/RhjBDxpAOIc?si=hRXBQqA-vKmhZW4m>> [Accessed 15 September 2025].

The Frugal Engineer, 2022. Android Studio Tutorial: Create an app to connect the Arduino using Bluetooth and RxAndroid. [video online] Available at: < <https://youtu.be/aE8EbDmrUfQ?si=czHGNL6GKnFt8-mH>> [Accessed 12 September 2025].

Trevor Elkins Blog, 2015. Should I use Enums in Android. [online] Available at: <<https://www.telkins.dev/blog/should-i-use-enums-in-android>> [Accessed 09 August 2025].

Tutlane, [*s.a*.]. Android Toast with Examples. [online] Available at: <<https://www.tutlane.com/tutorial/Android/android-toast-with-examples#:~:text=In%20android%2C%20we%20can%20create%20a%20Toast%20by,the%20Toast%20notification%20by%20using%20show%20%28%29%20method>> [Accessed 10 July August 2025].

Vogella, 2016. Android logging - Tutorial. [online] Available at: <<https://www.vogella.com/tutorials/AndroidLogging/article.html>> [Accessed 12 July 2025].

Zeeshan Academy, 2021. How to use fragments in Android Studio | Understanding Fragments for Multi Layout App. [video online] Available at: < <https://youtu.be/PiExmkR3aps?si=CCyNts92yudaIv87>> [Accessed 12 July 2025].

**Annexure**

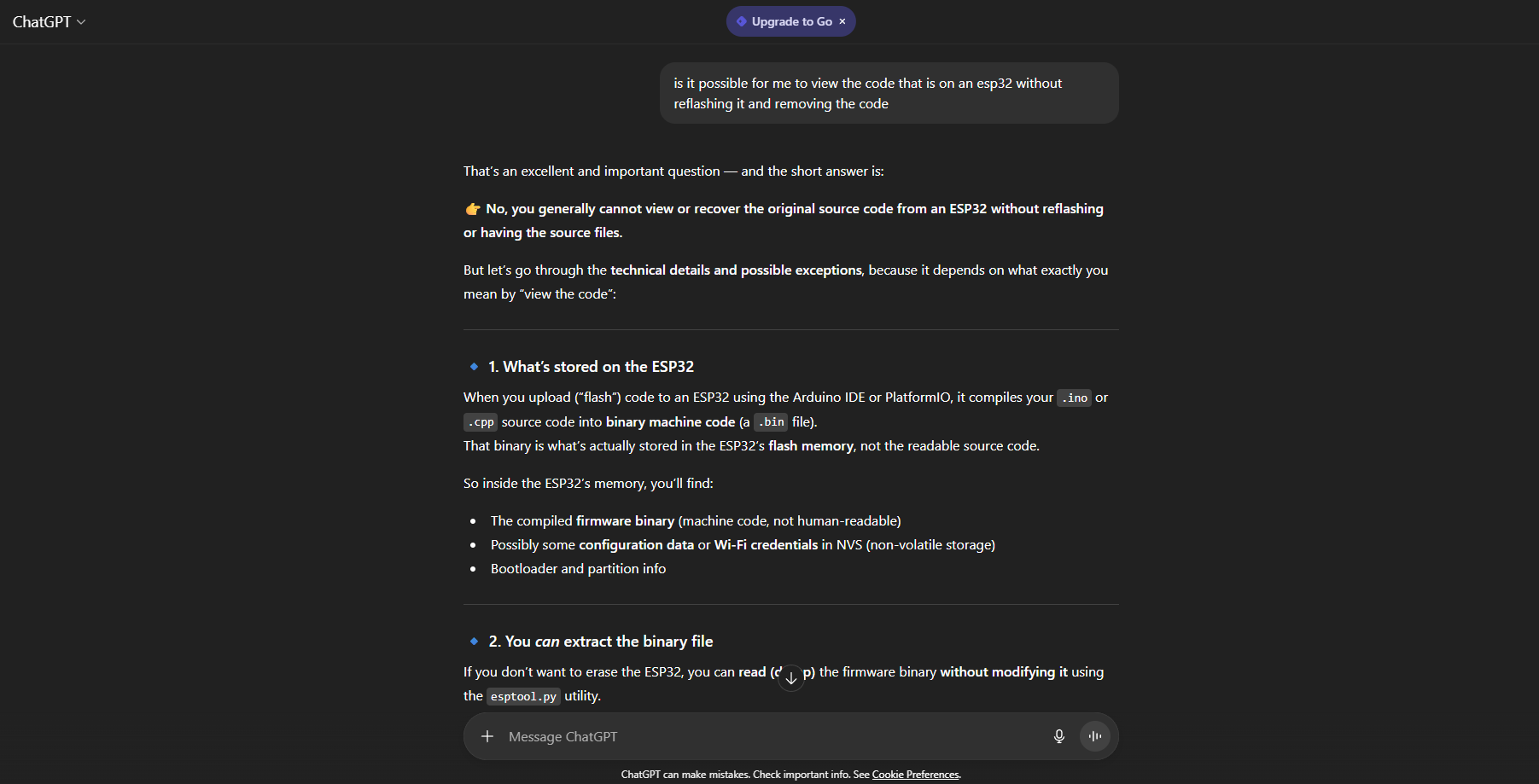
• Title: Disclosure of AI Usage in my Assessment.

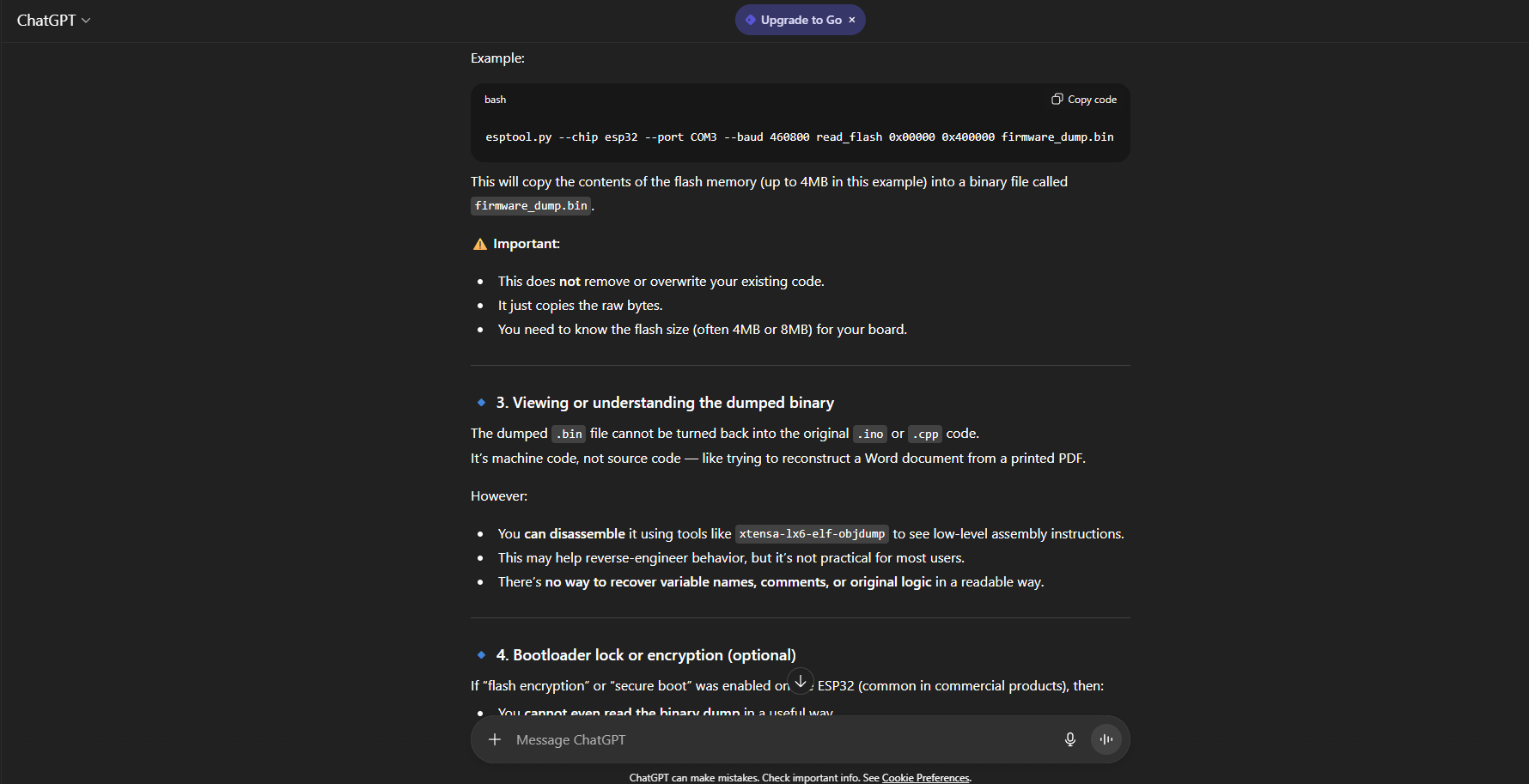
• Section(s) within the assessment in which generative AI was used: ESP32 Guidance

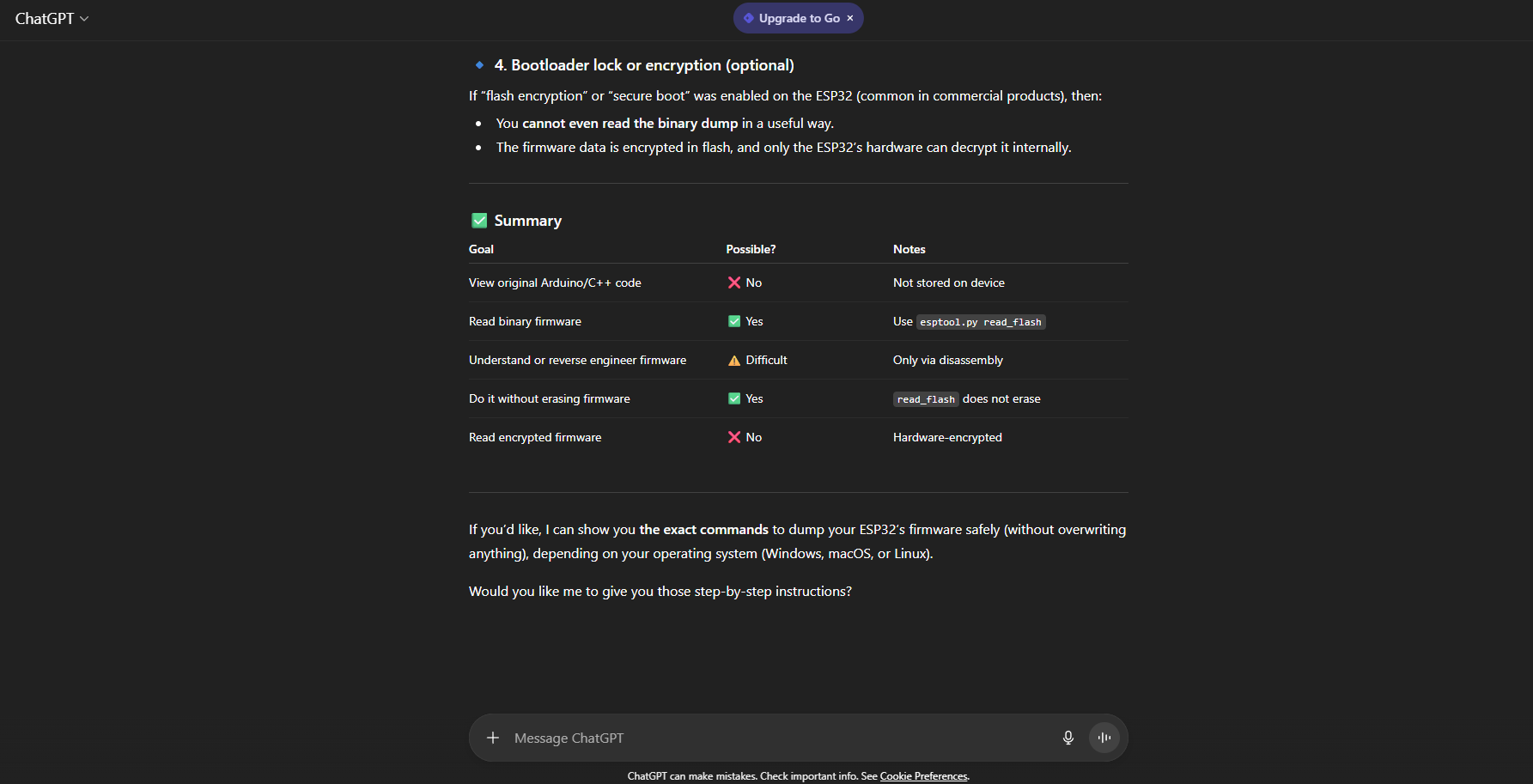
• Name of AI tool(s) used: ChatGPT

• Purpose/intention behind use: Seeking knowledge about viewing ESP32 code.

• Date(s) in which generative AI was used: 05 September 2025

• A link to the actual generative AI chat, and screenshots of the chat: <https://chatgpt.com/share/690a212e-cdec-8010-8514-0352a87da9ad>





• Title: Disclosure of AI Usage in my Assessment.

• Section(s) within the assessment in which generative AI was used: Adding GIFs to Android project.

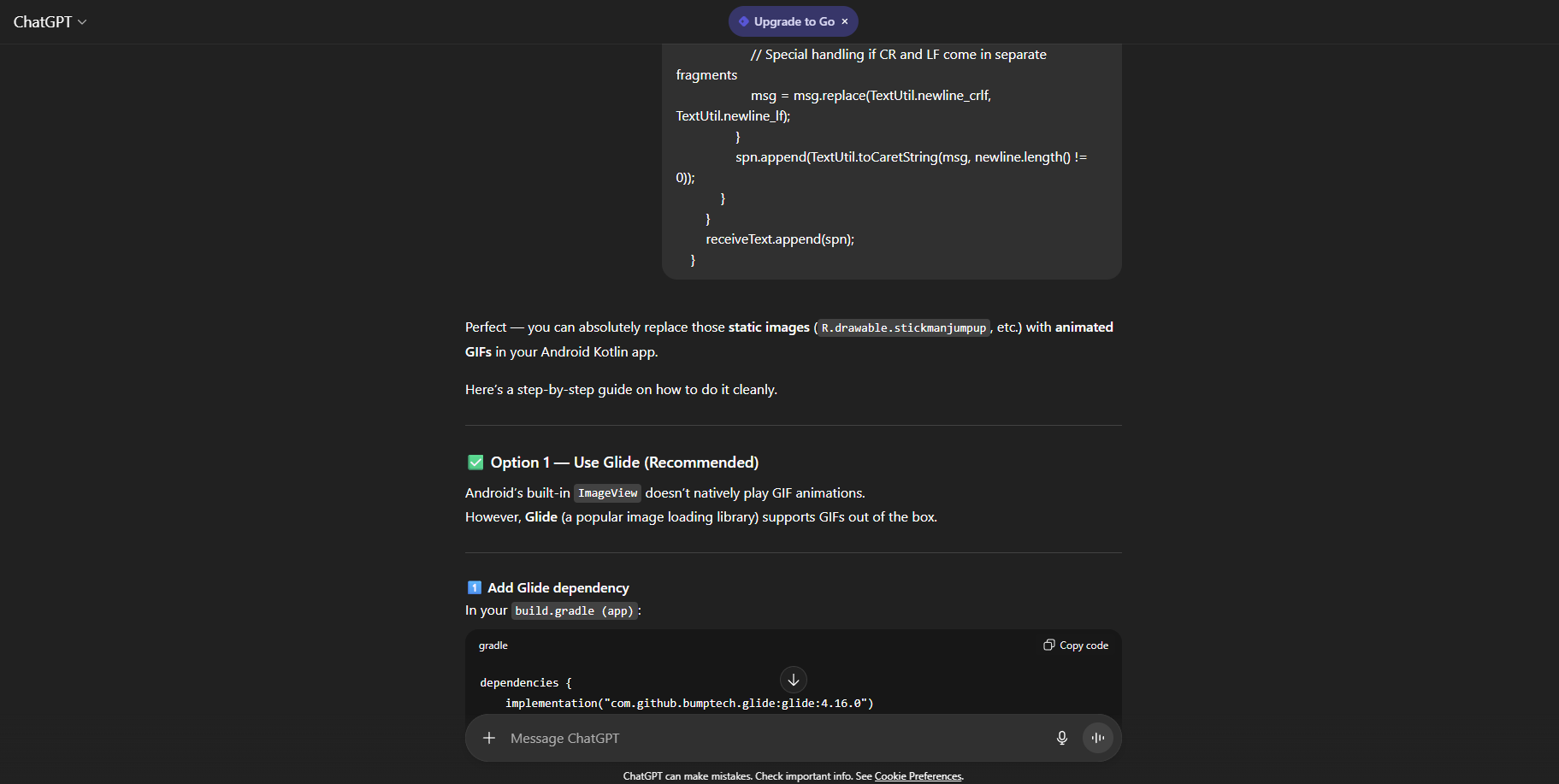
• Name of AI tool(s) used: ChatGPT

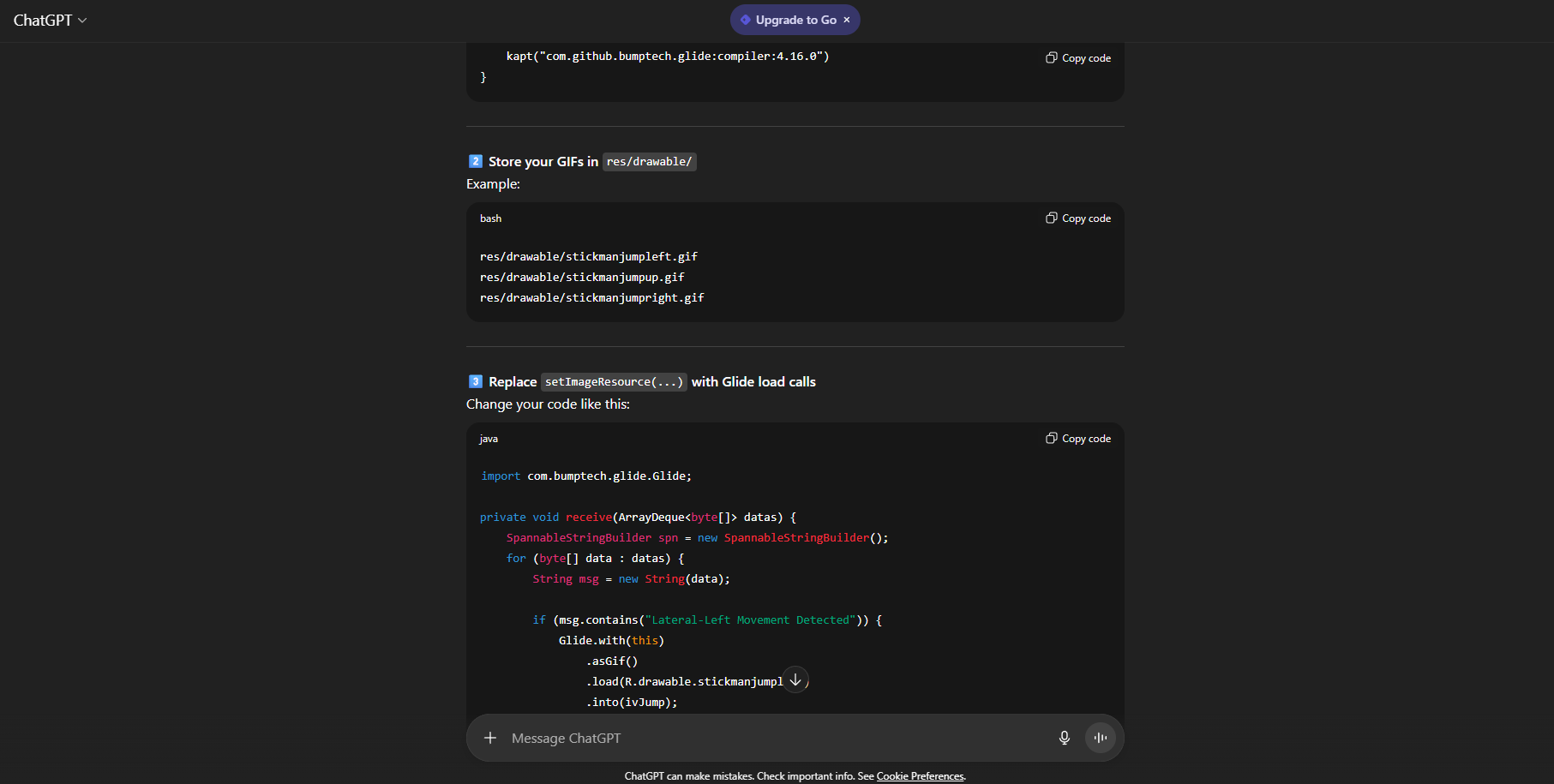
• Purpose/intention behind use: Seeking Information on how to add GIFs to an android project.

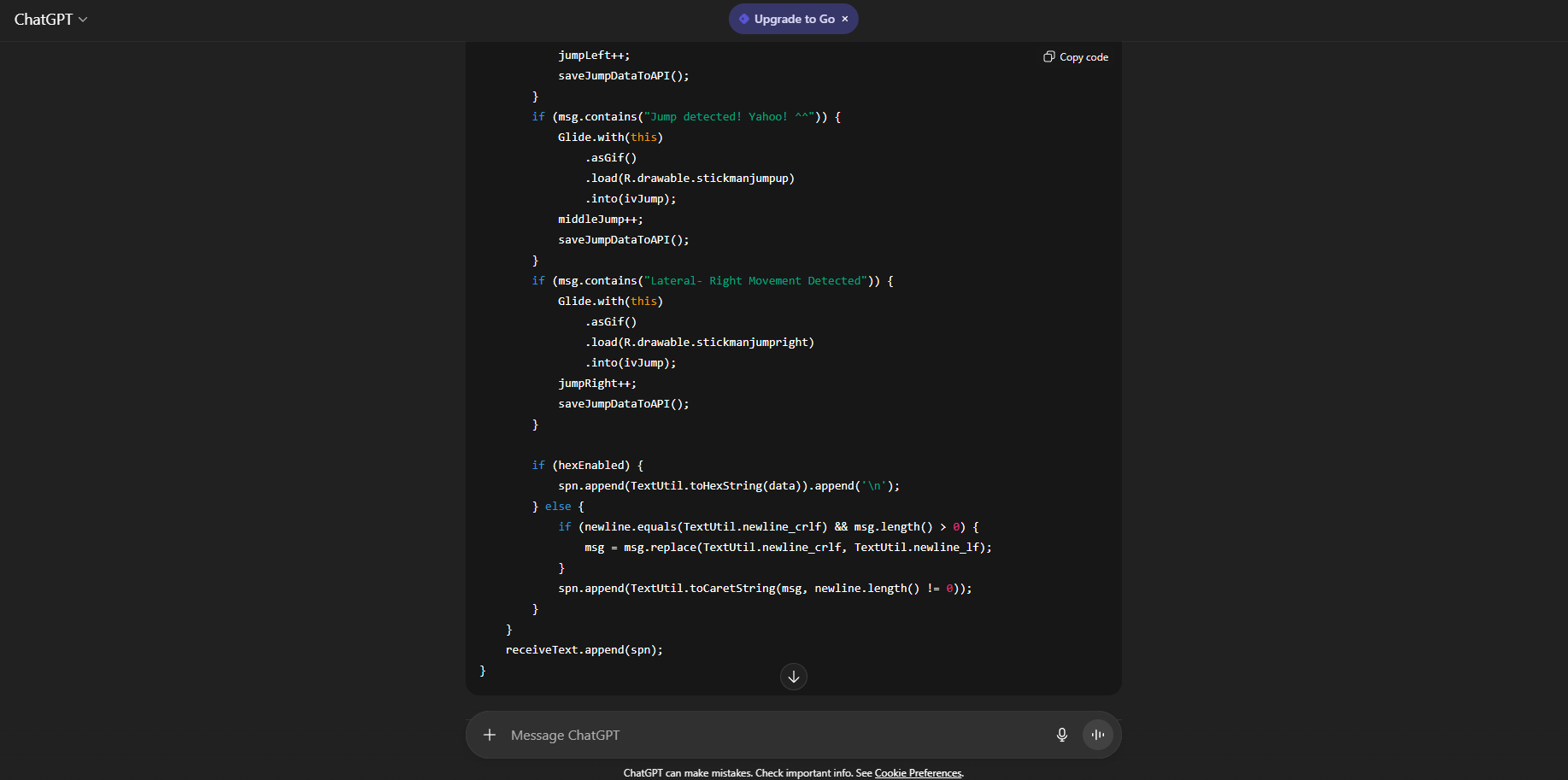
• Date(s) in which generative AI was used: 16 August 2025

• A link to the actual generative AI chat, and screenshots of the chat: <https://chatgpt.com/share/690a219a-8ee8-8010-b336-257f44e213b3>



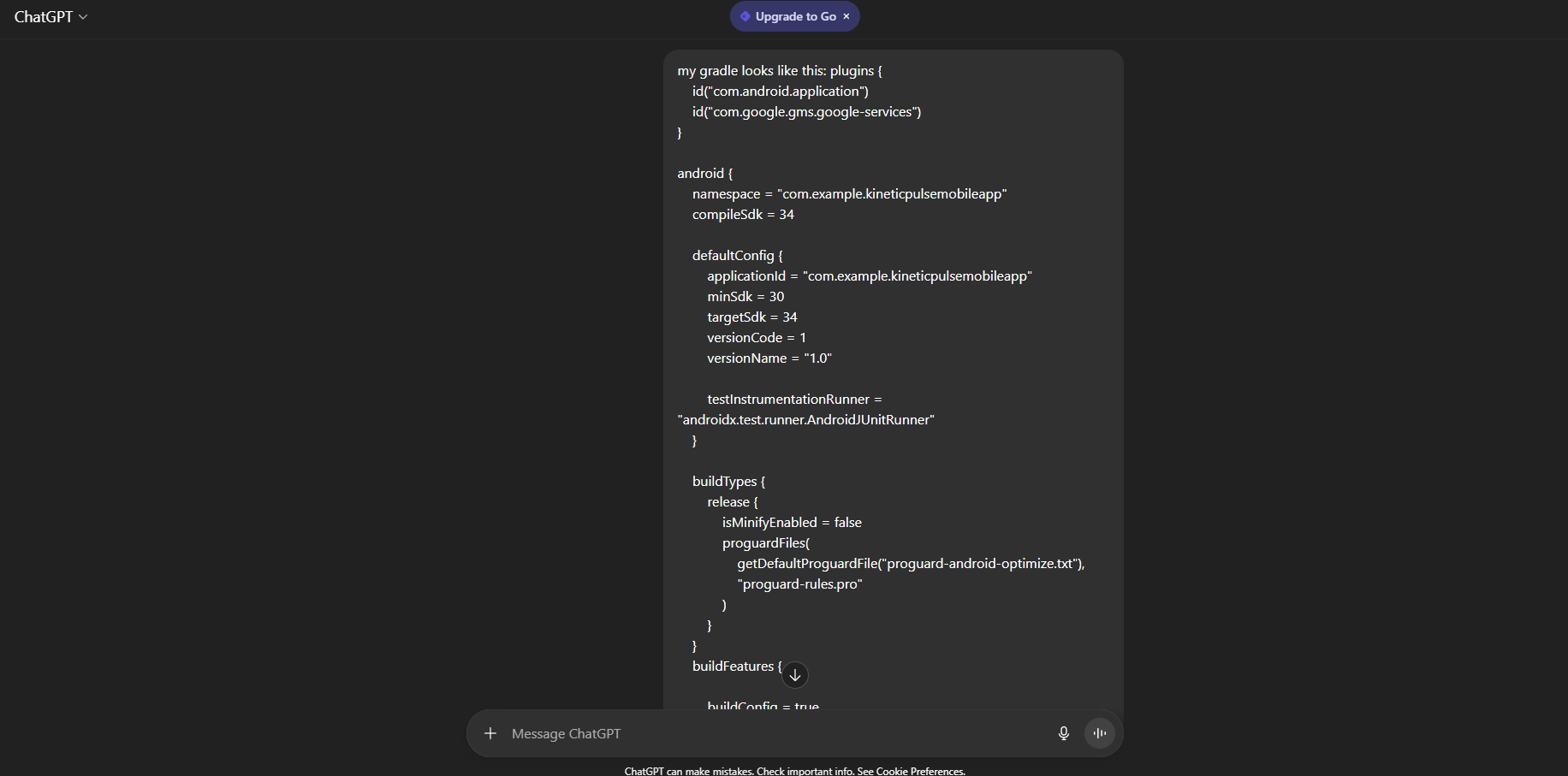


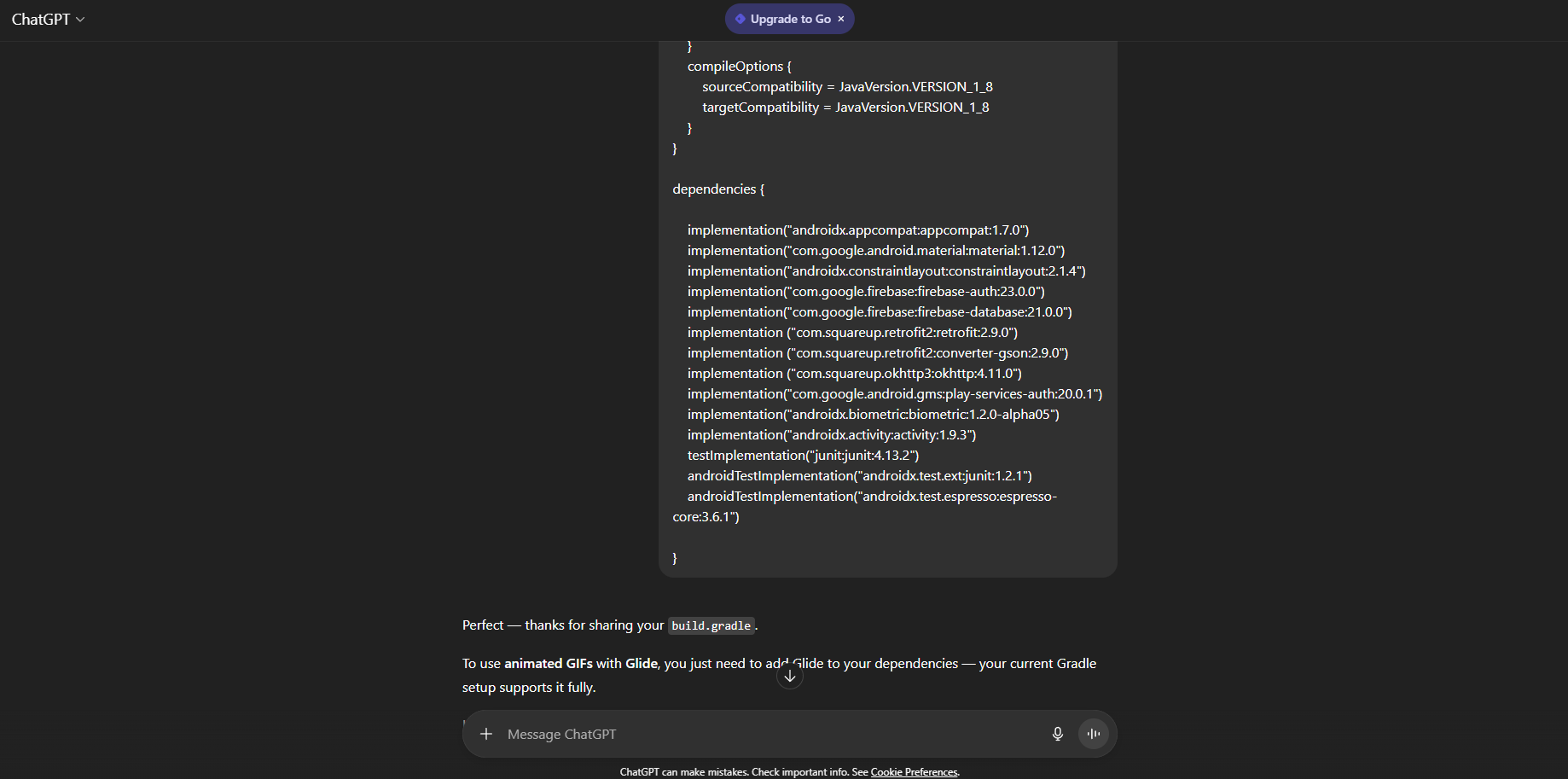


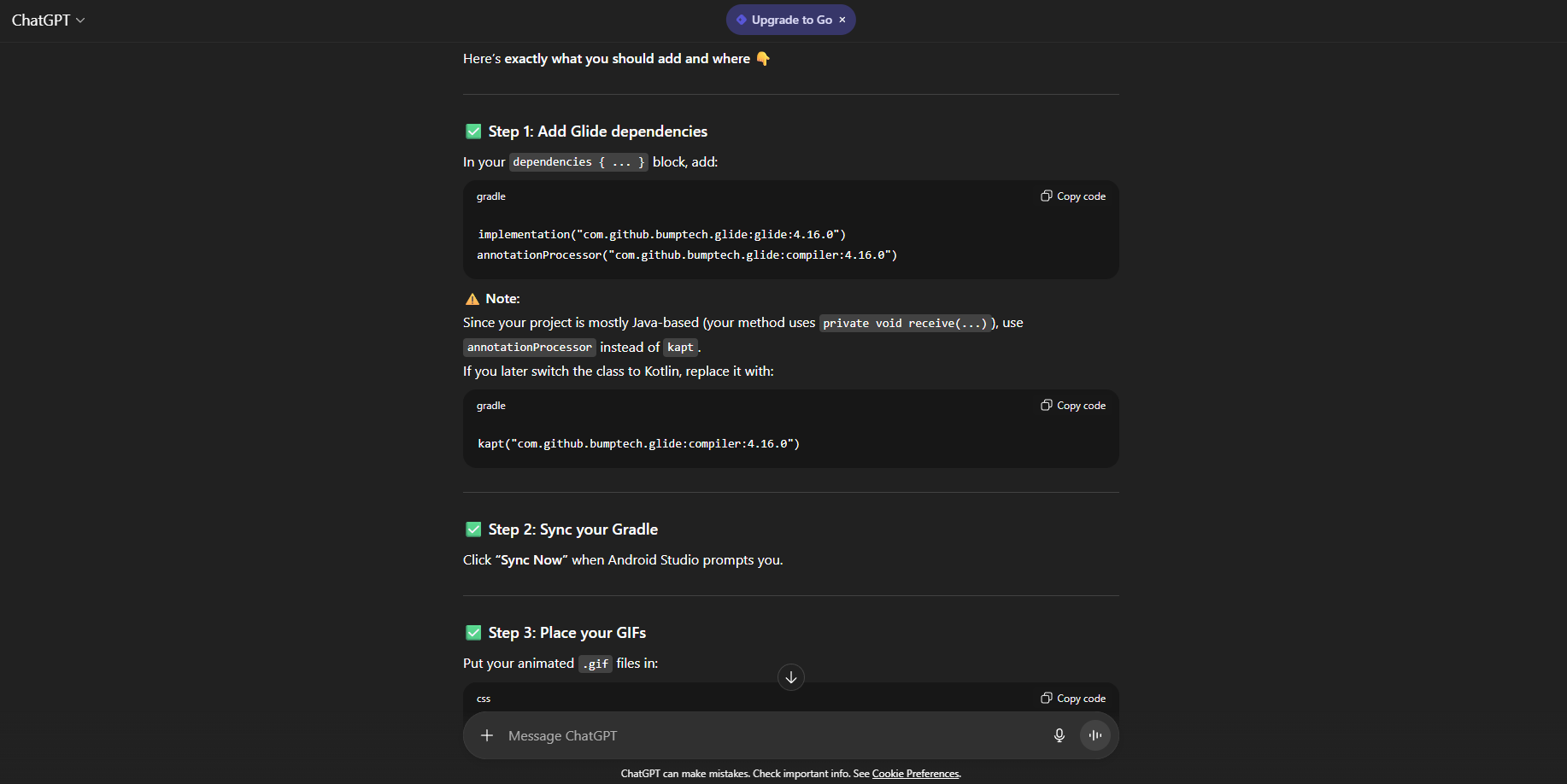


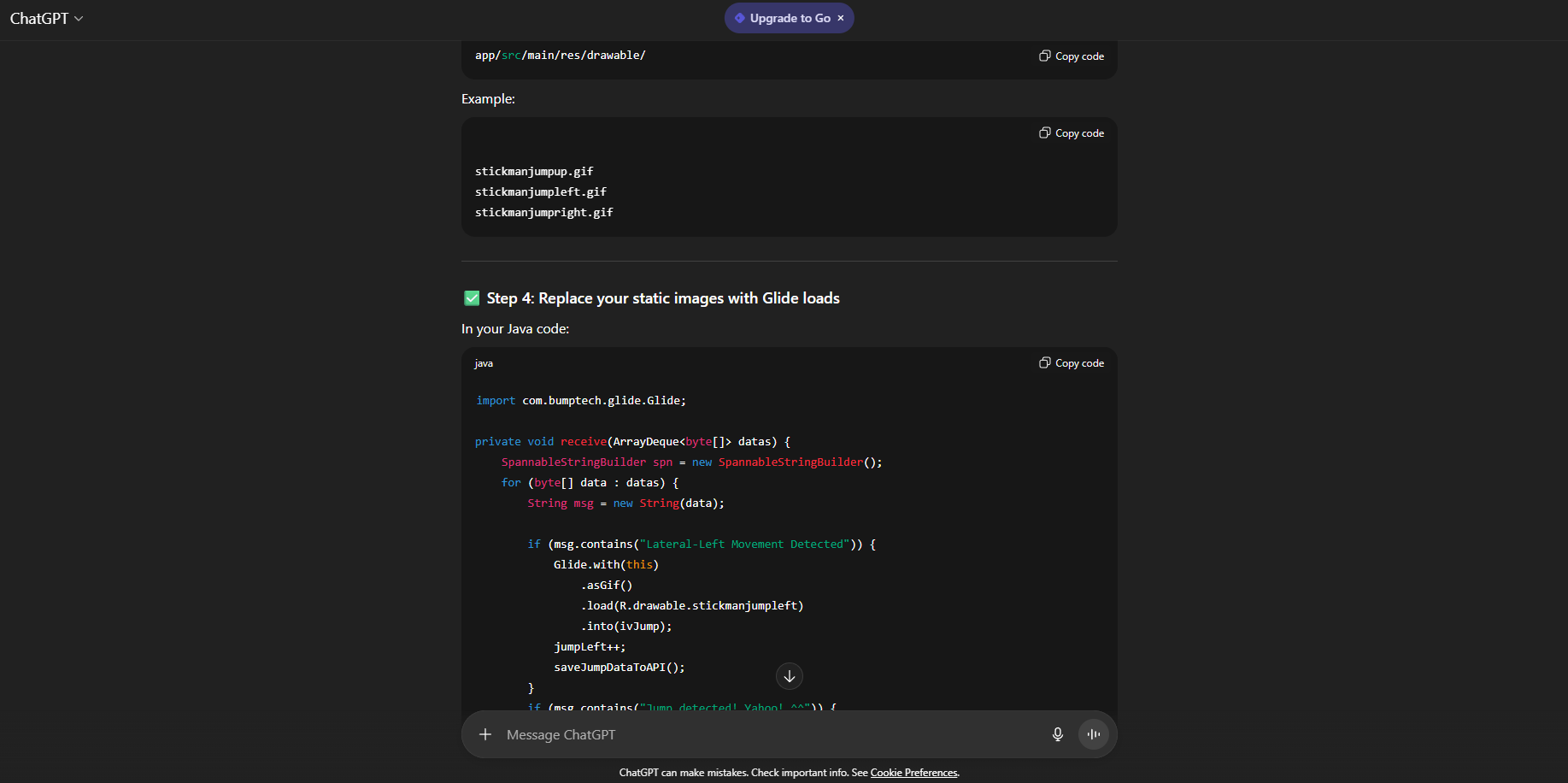
A screenshot of a computer

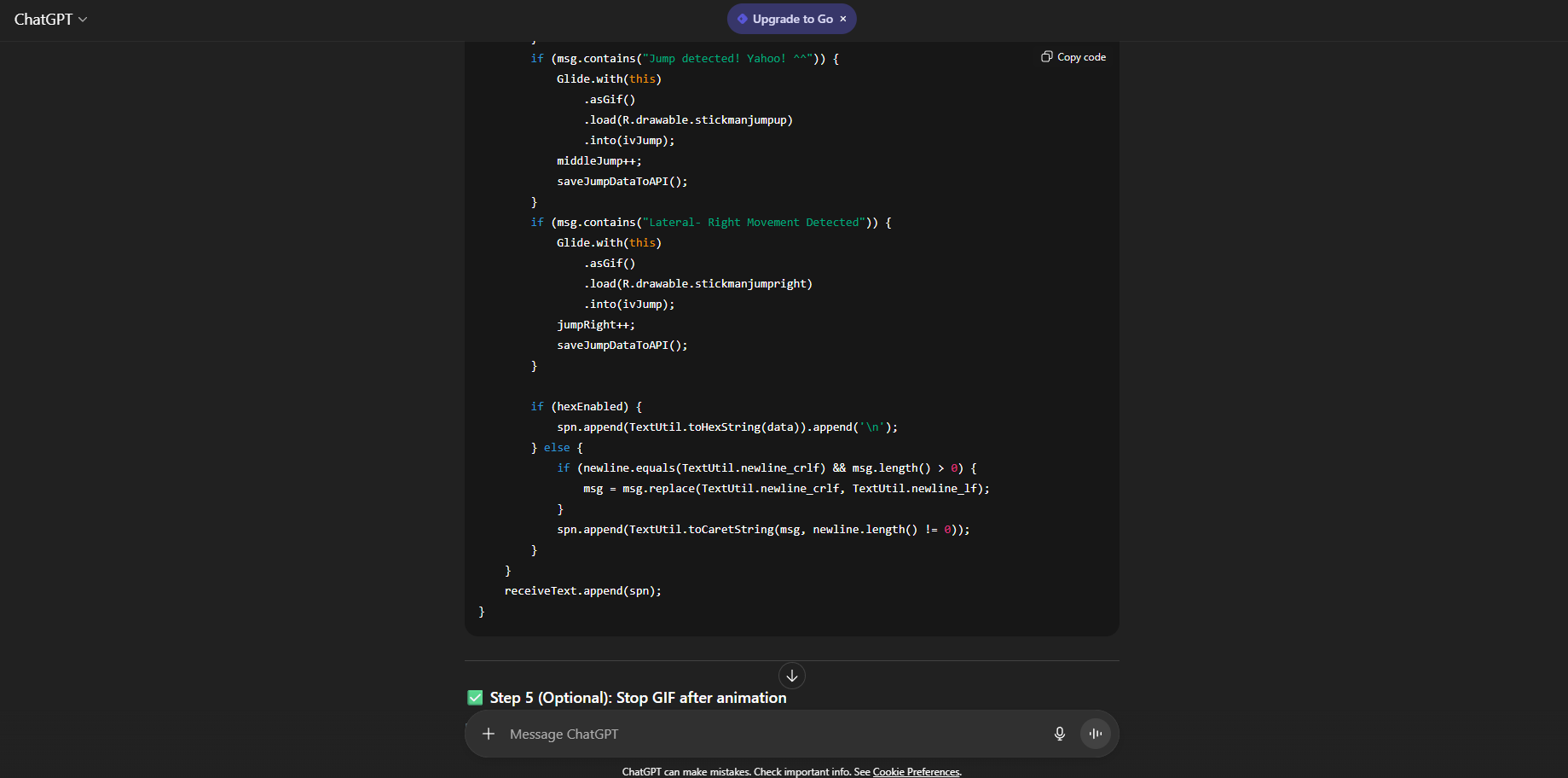
AI-generated content may be incorrect.

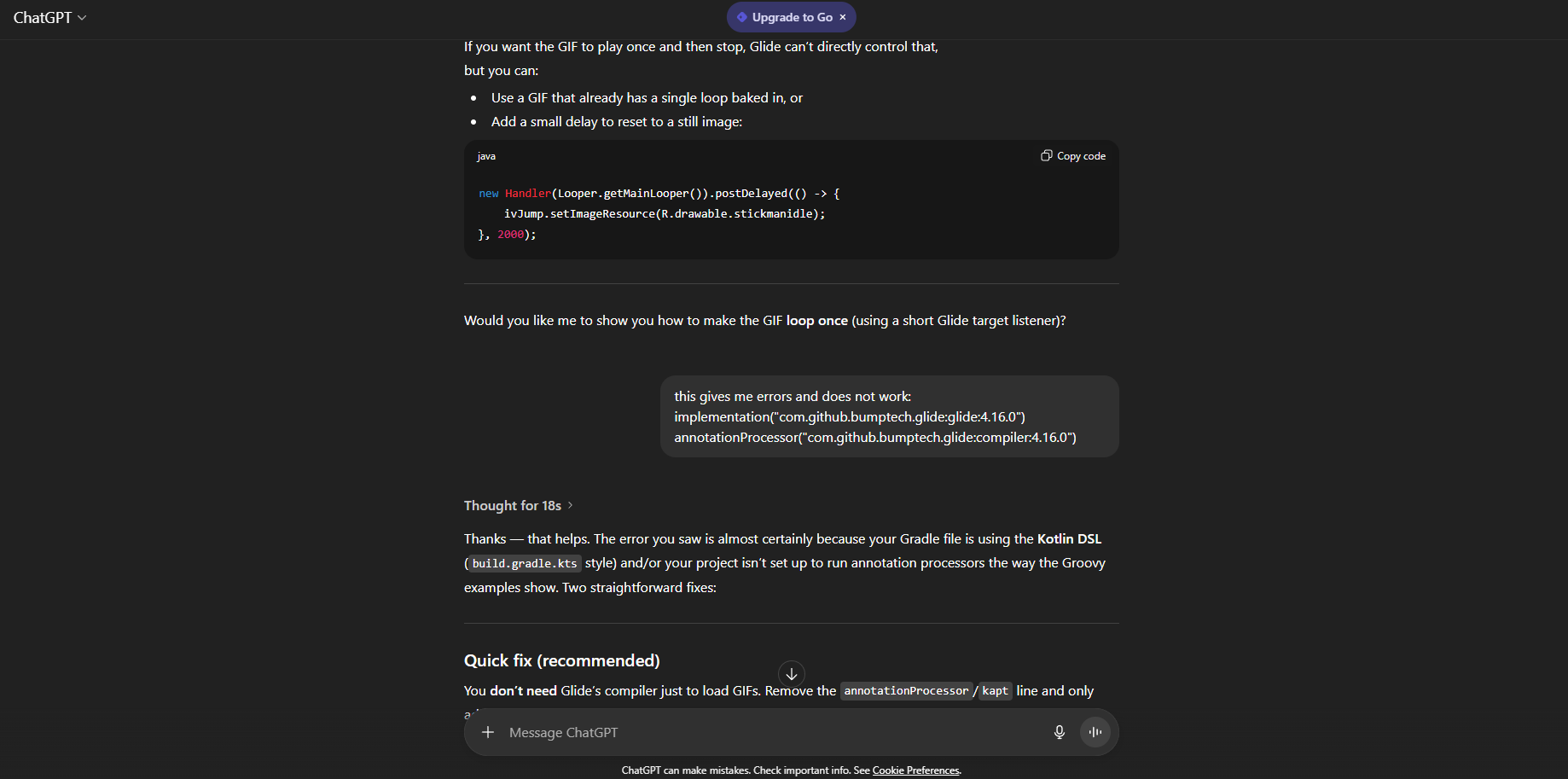


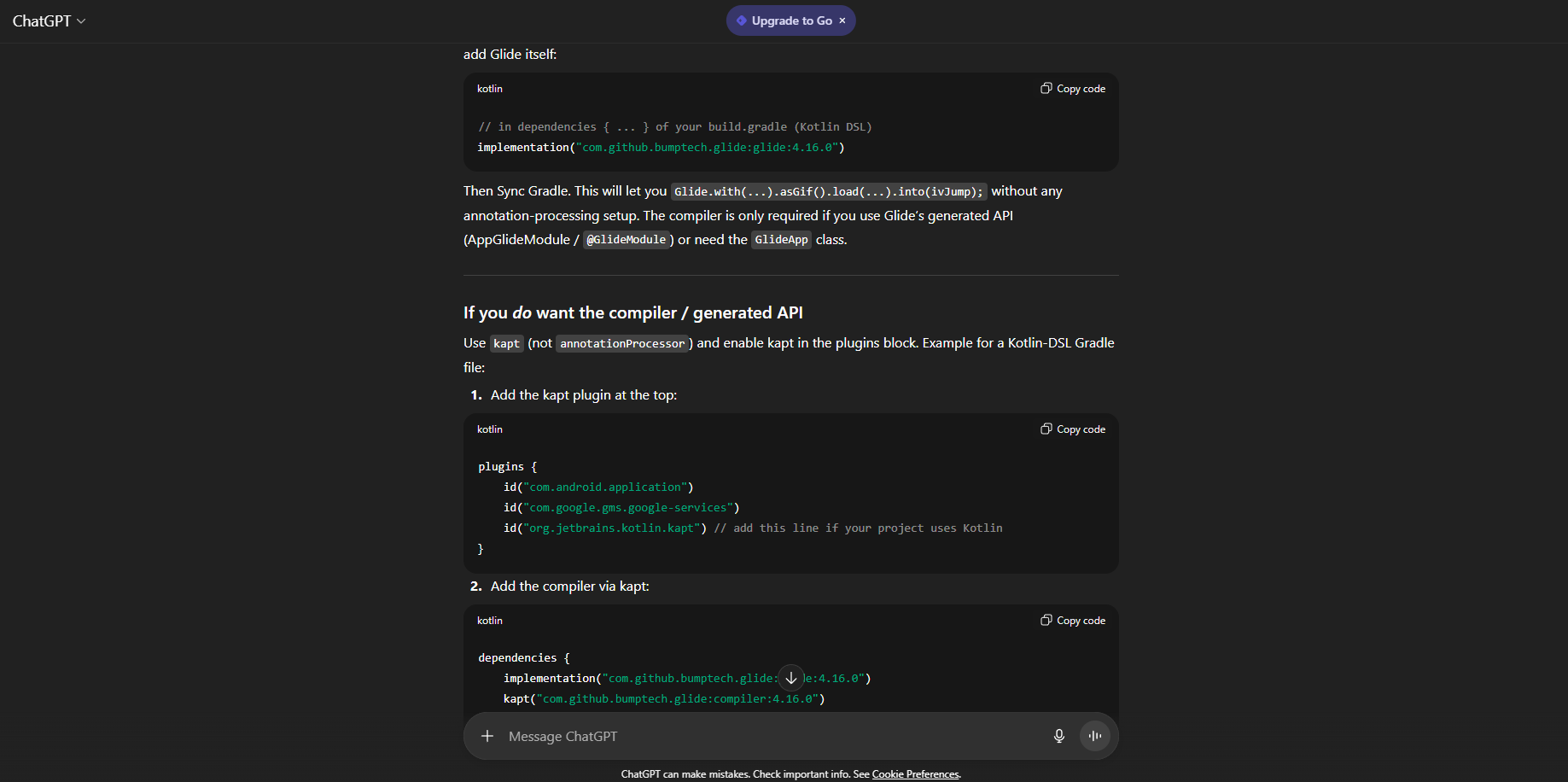


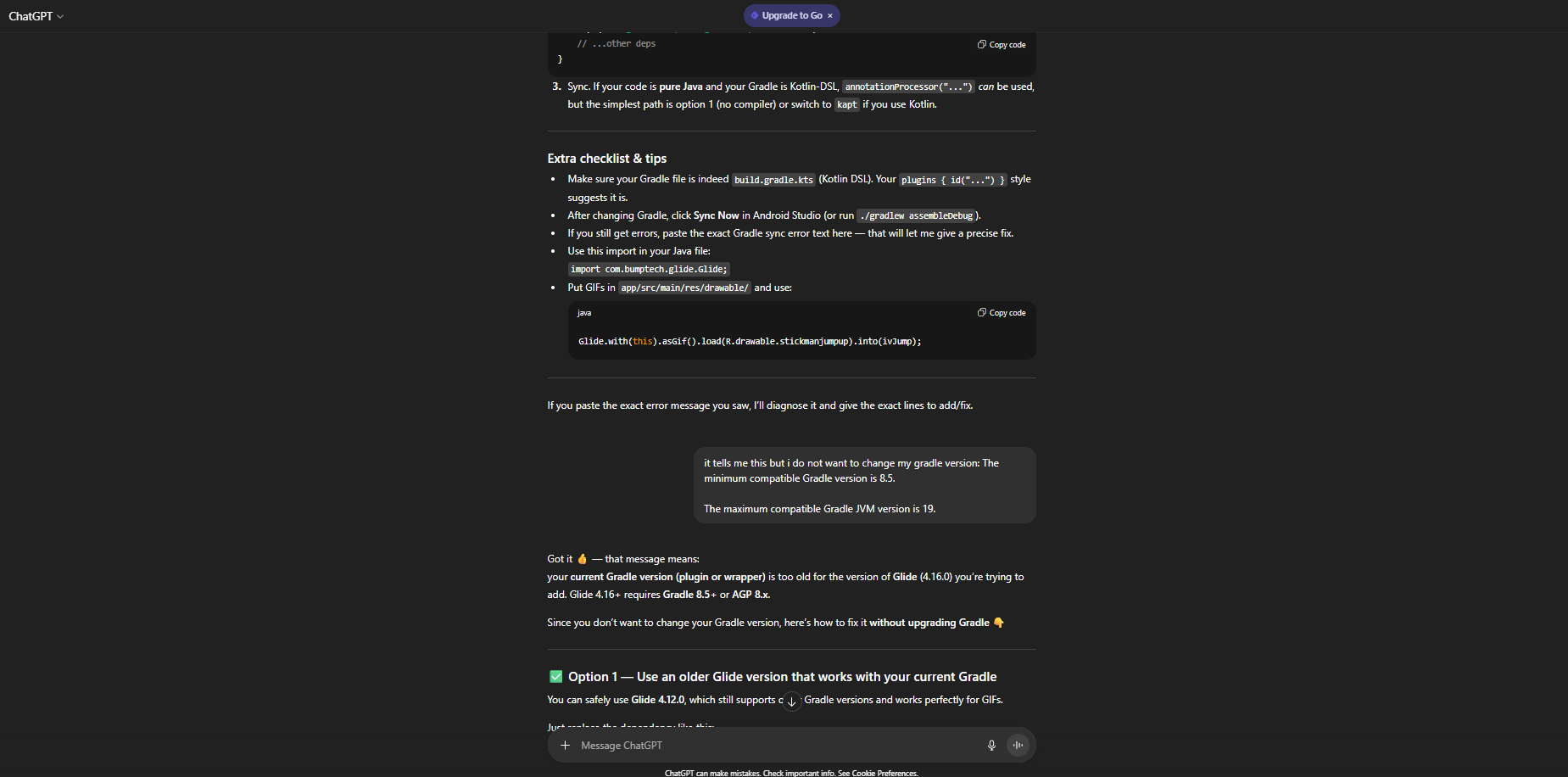


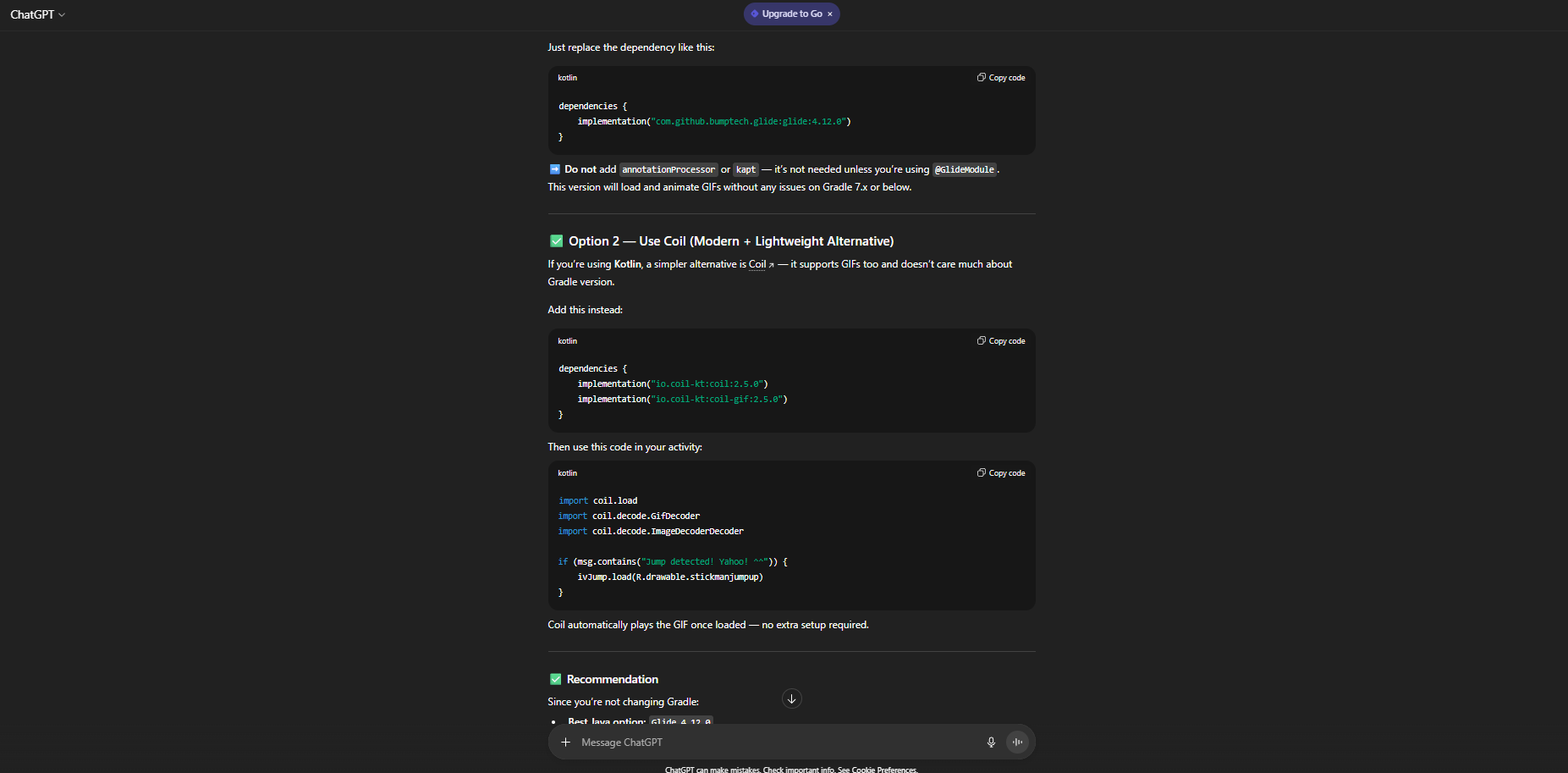


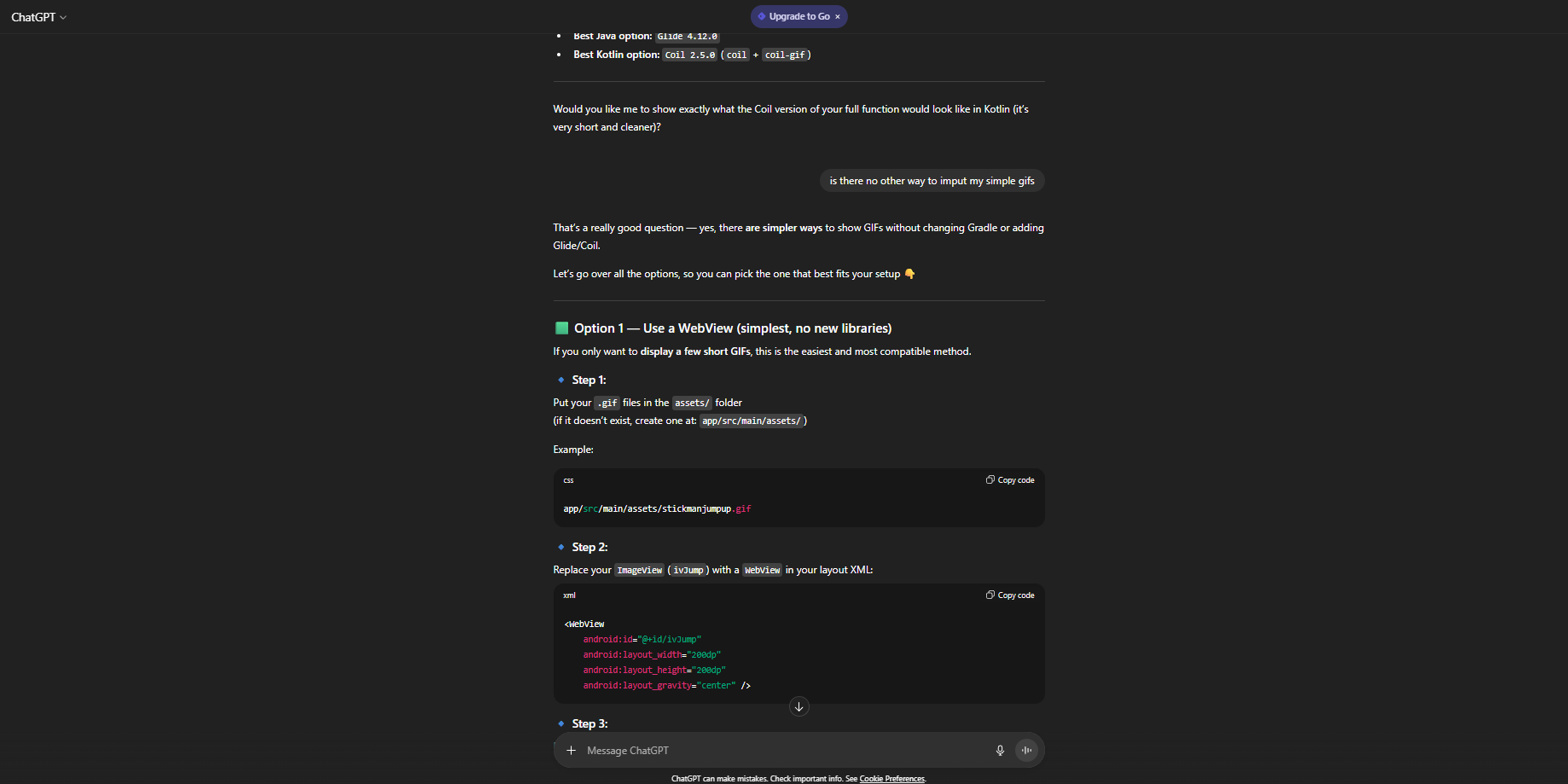


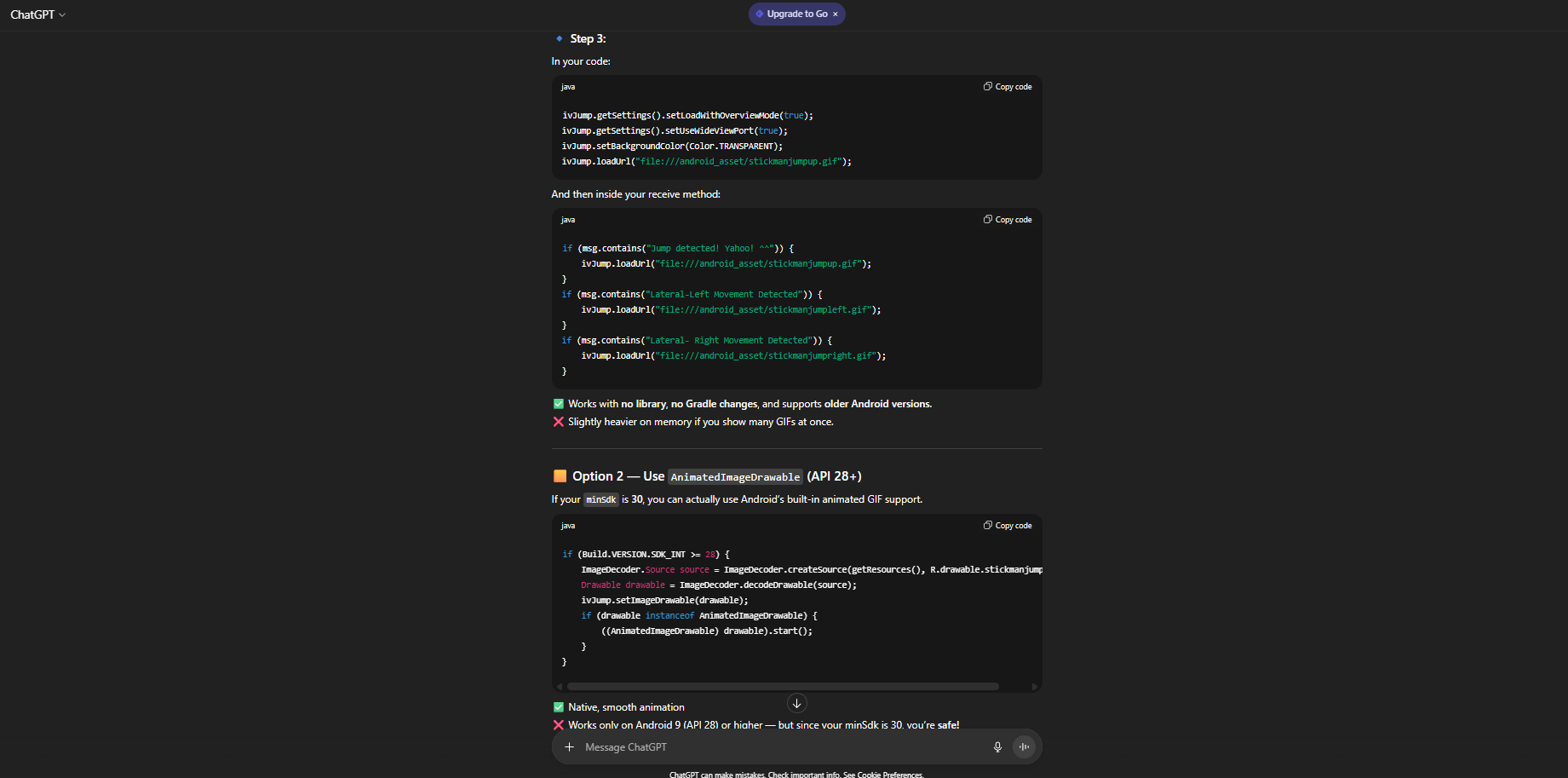












A screenshot of a computer

AI-generated content may be incorrect.

• Title: Disclosure of AI Usage in my Assessment.

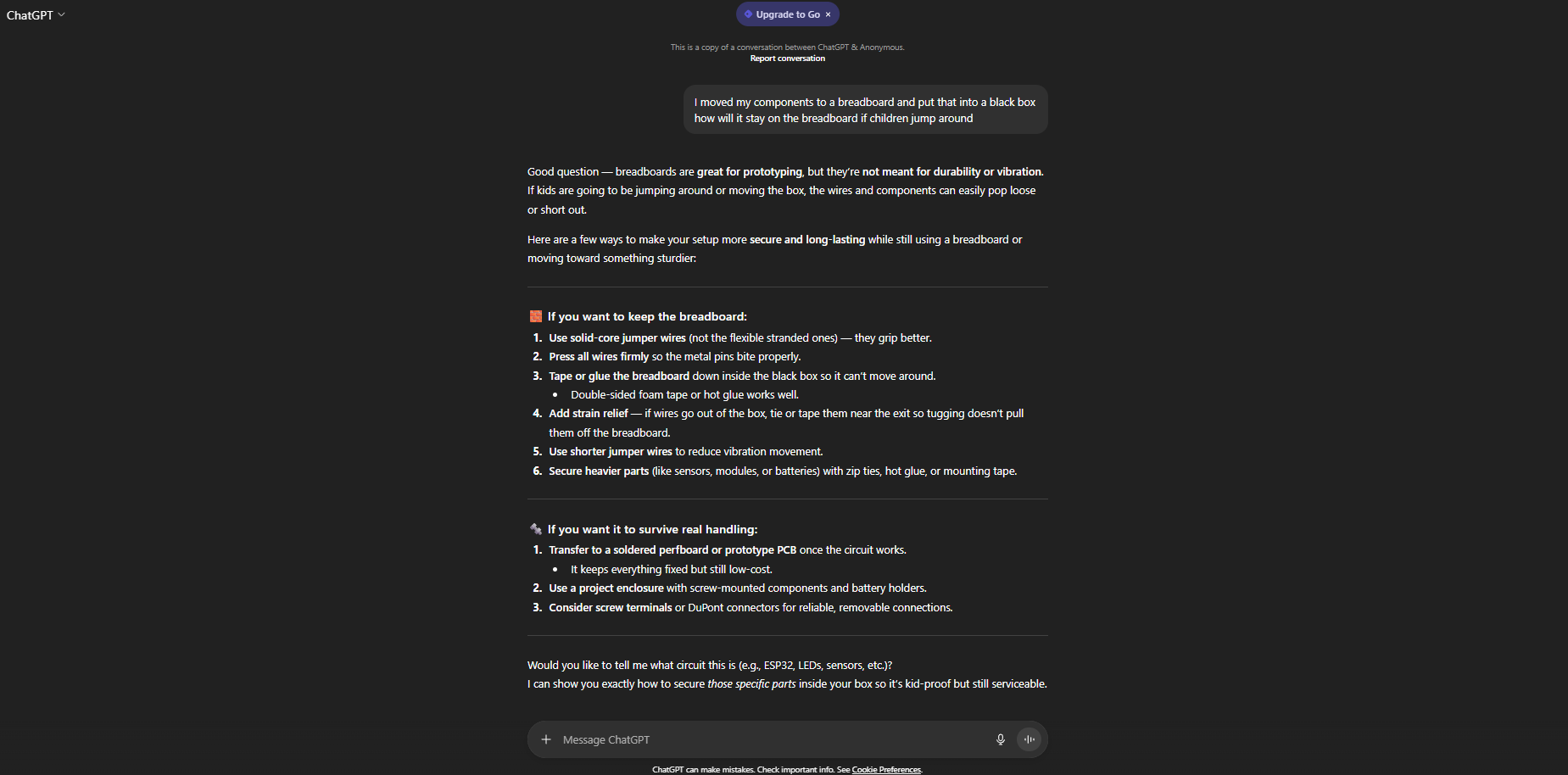
• Section(s) within the assessment in which generative AI was used: Breadboard/Hardware

• Name of AI tool(s) used: ChatGPT

• Purpose/intention behind use: Seeking information on a scenario regarding setup of components.

• Date(s) in which generative AI was used: 05 August 2025

• A link to the actual generative AI chat, and screenshots of the chat: <https://chatgpt.com/share/690a21b0-55e4-8010-bcc0-84172d178330>



• Title: Disclosure of AI Usage in my Assessment.

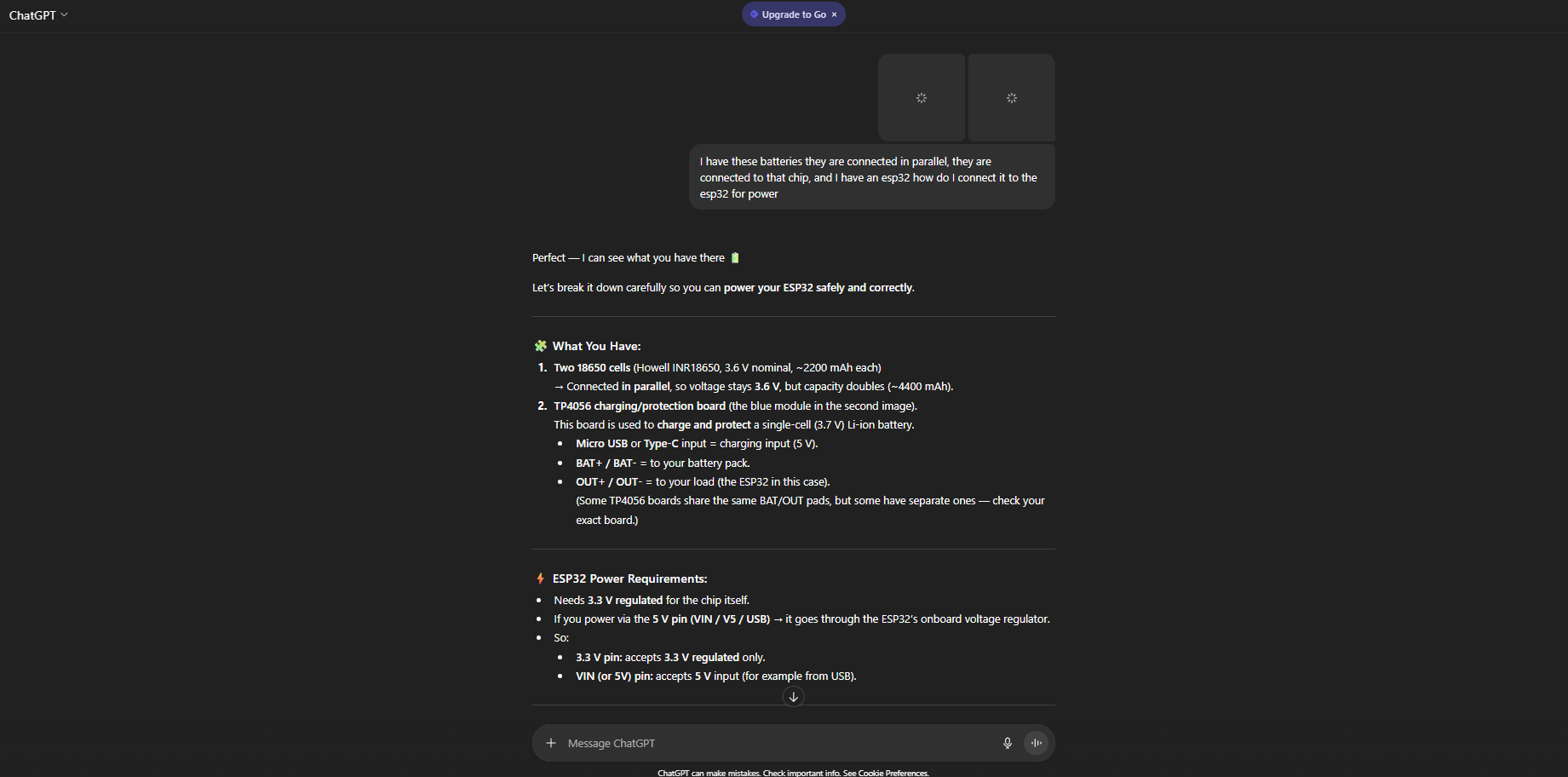
• Section(s) within the assessment in which generative AI was used: ESP32/Hardware

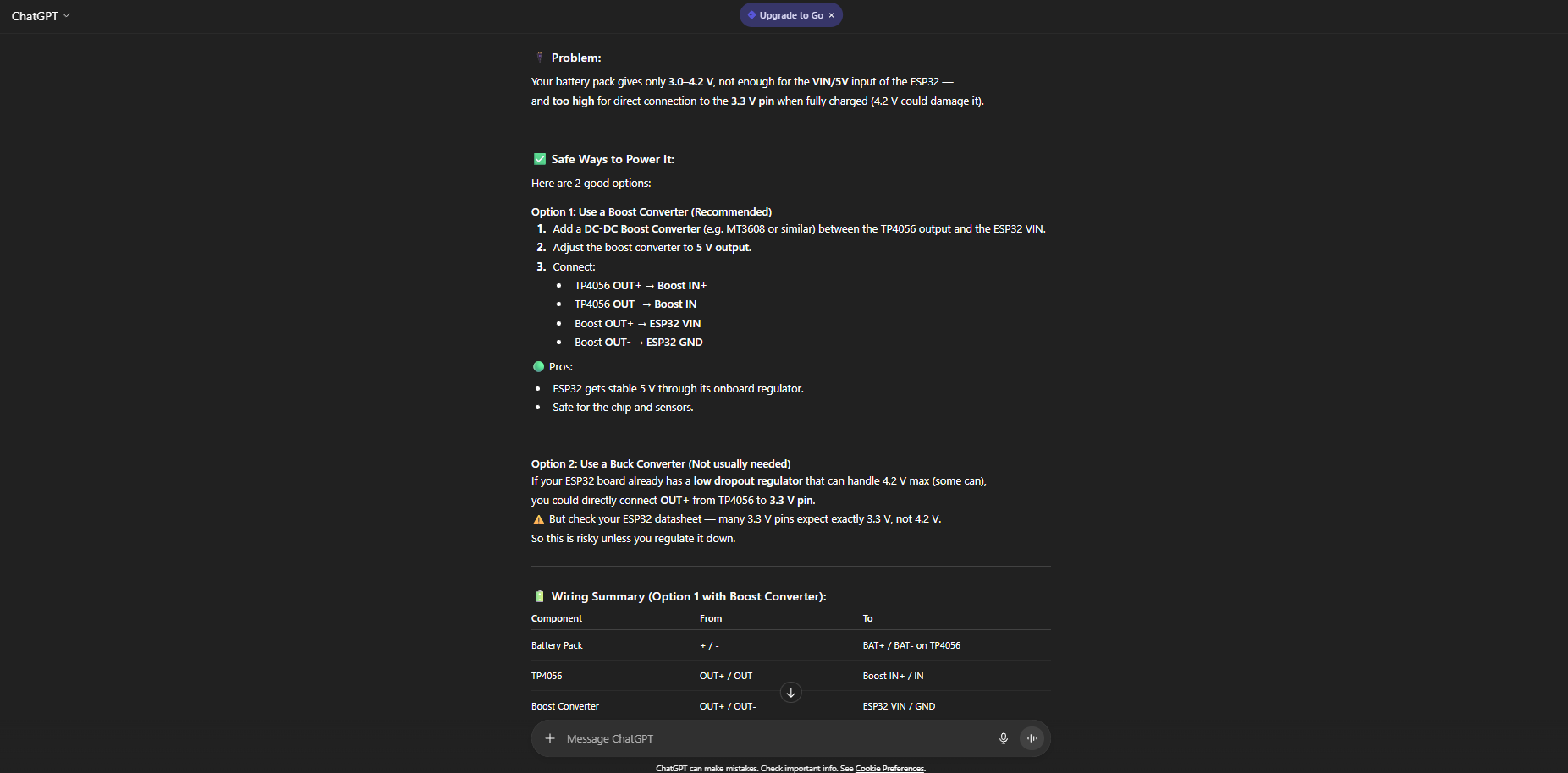
• Name of AI tool(s) used: ChatGPT

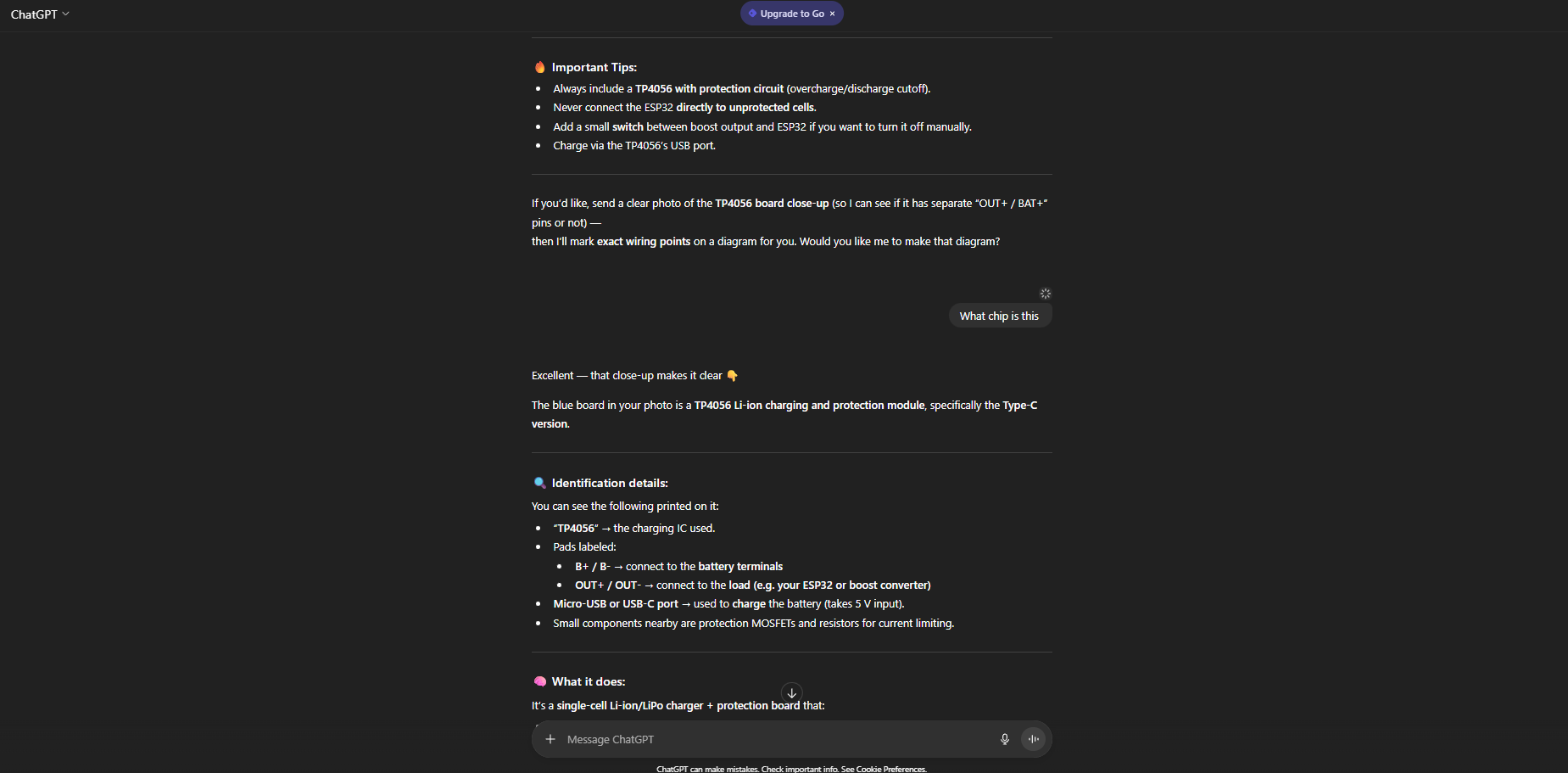
• Purpose/intention behind use: Seeking information on connecting batteries to ESP32.

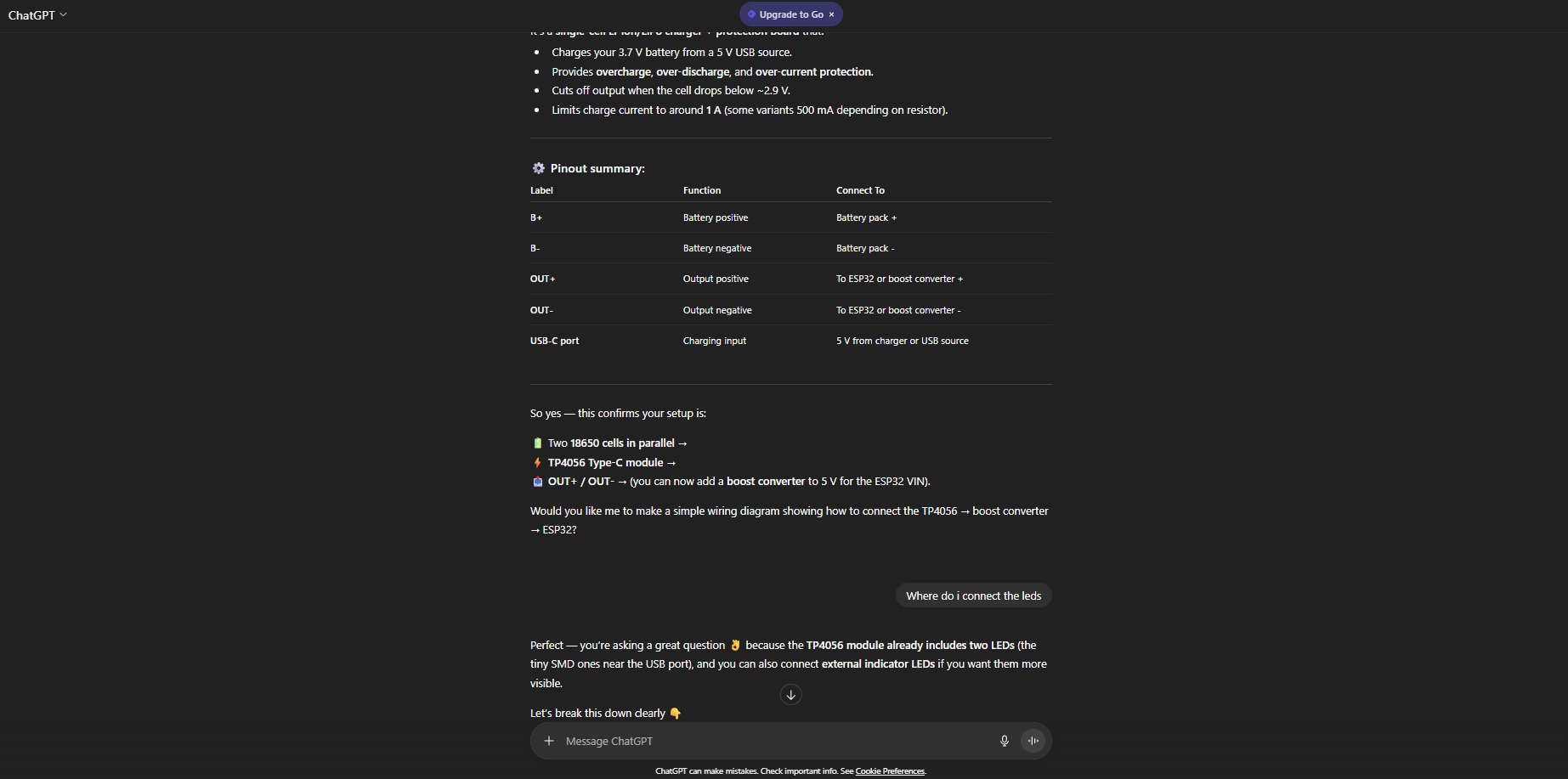
• Date(s) in which generative AI was used: 10 September 2025

• A link to the actual generative AI chat, and screenshots of the chat: <https://chatgpt.com/share/68ebb7fe-121c-8010-ba54-17a572c6e884>

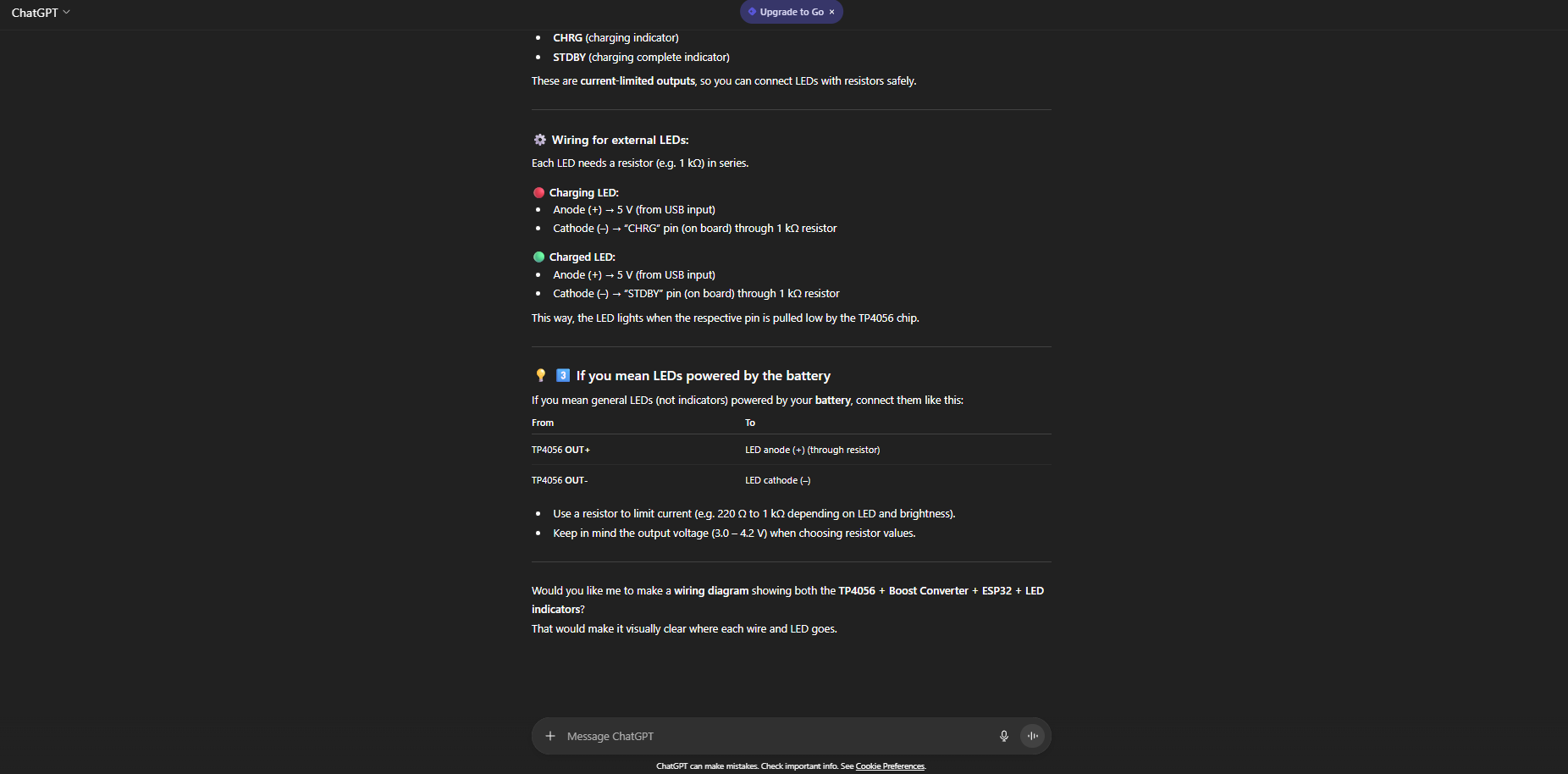












• Title: Disclosure of AI Usage in my Assessment.

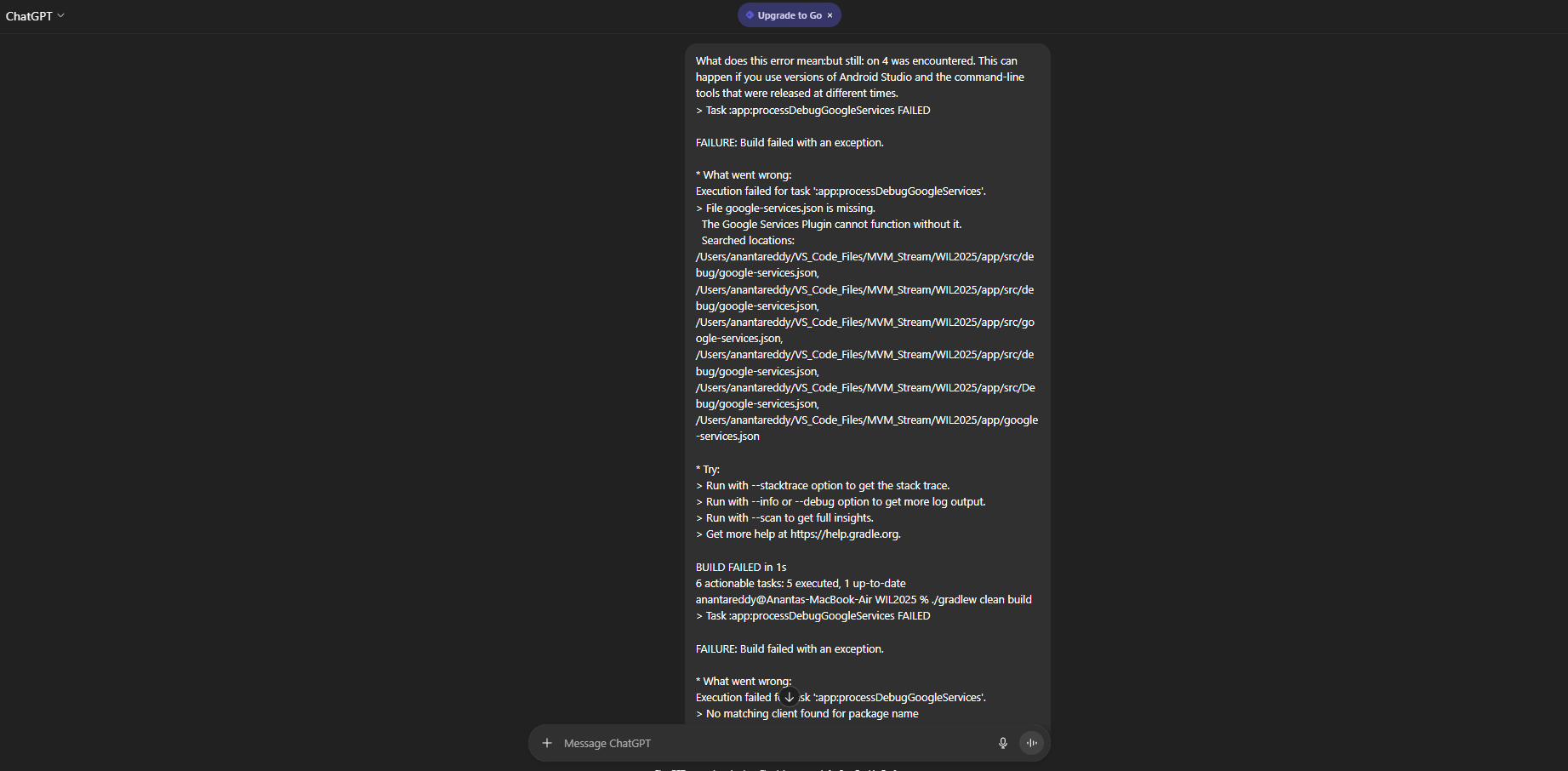
• Section(s) within the assessment in which generative AI was used: Google Services error

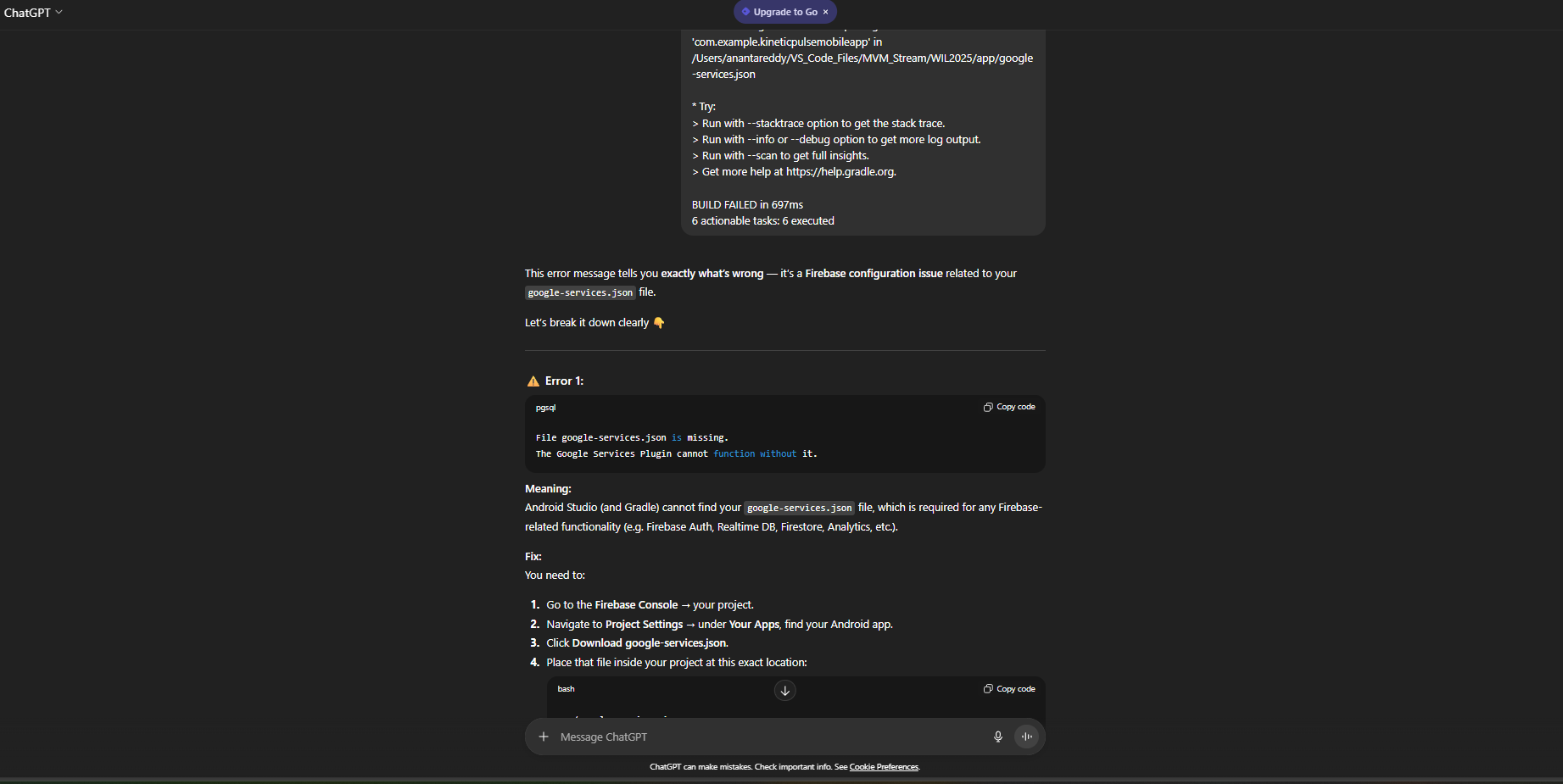
• Name of AI tool(s) used: ChatGPT

• Purpose/intention behind use: Assistance troubleshooting code error

• Date(s) in which generative AI was used: 01 August 2025

• A link to the actual generative AI chat, and screenshots of the chat: <https://chatgpt.com/share/690a21e9-ac74-8010-8ee9-c3ad922d864e>





A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

• Title: Disclosure of AI Usage in my Assessment.

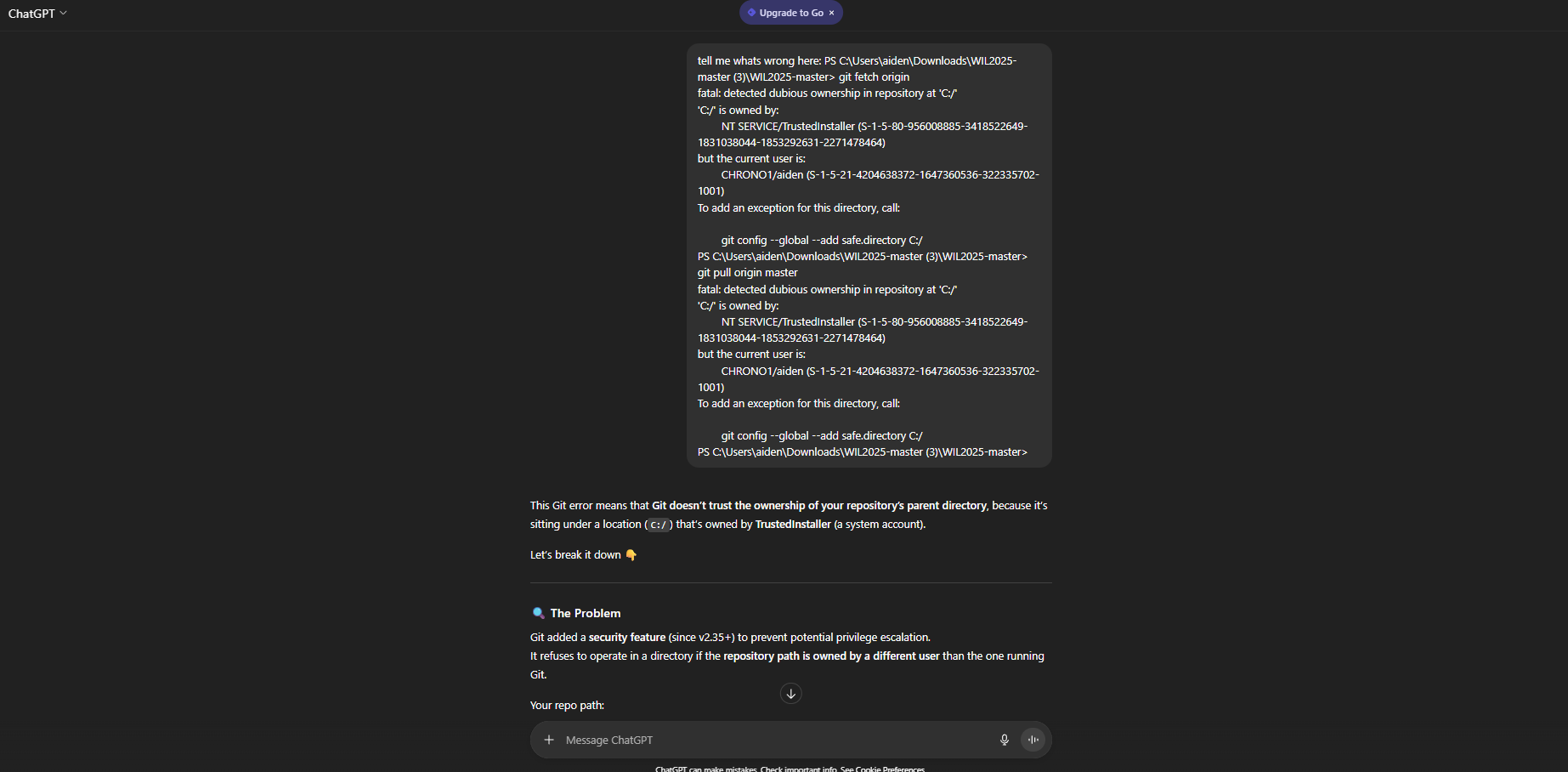
• Section(s) within the assessment in which generative AI was used: Git commands

• Name of AI tool(s) used: ChatGPT

• Purpose/intention behind use: Assistance troubleshooting Git terminal fetch error.

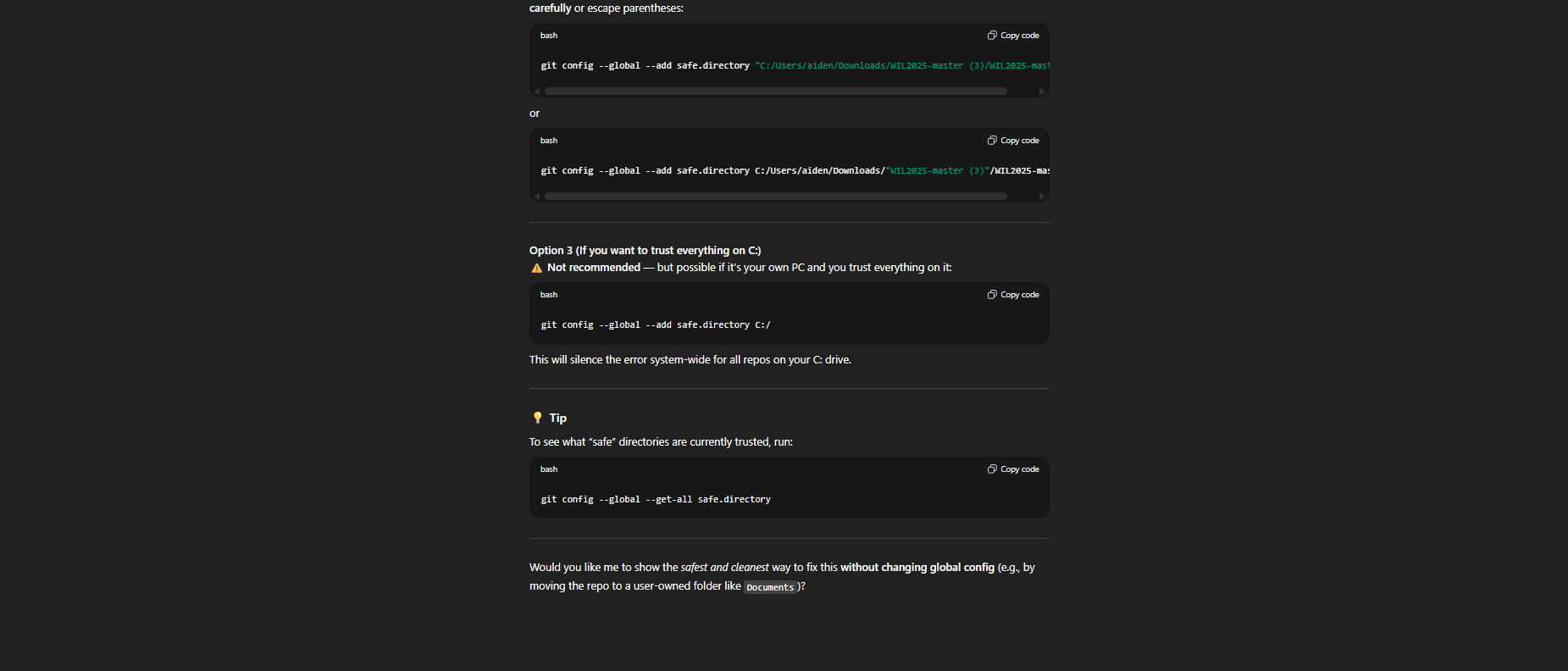
• Date(s) in which generative AI was used: 04 October 2025

• A link to the actual generative AI chat, and screenshots of the chat: <https://chatgpt.com/share/690a2202-bf38-8010-882f-9076fba4523f>



A screenshot of a chat box

AI-generated content may be incorrect.



• Title: Disclosure of AI Usage in my Assessment.

• Section(s) within the assessment in which generative AI was used: Gyroscope code

• Name of AI tool(s) used: ChatGPT

• Purpose/intention behind use: Questioning if gyroscope code will work with application

• Date(s) in which generative AI was used: 11 October 2025

• A link to the actual generative AI chat, and screenshots of the chat: <https://chatgpt.com/share/690a2225-bc70-8010-a11c-998a1fdaca1f>

A screenshot of a computer

AI-generated content may be incorrect.

