CODEX MERCHANTS

COS 301 | CAPSTONE PROJECT | UNIVERSITY OF PRETORIA

Gesture-Based Drone Control

EPI-USE®

CODEX MERCHANTS

Name	Student Number
Ayush Beekum	u23596351
Jaitin Moodally	u23621372
Shavir Vallabh	u23718146
Diya Narotam	u23533596
Chinmayi Santhosh	u24585671

Team email: codexmerchants@gmail.com

Introduction

Most drones today are operated using a traditional controller of some form, with analogue sticks, buttons or through proprietary mobile applications. While effective, these mechanisms require specialized hardware familiarity and impose a steep learning curve for novices. This limits accessibility, reduces intuitiveness, and created barriers for broader adoption in educational and experiential environments.

Traditional drone control also constrains natural human-computer interaction. Existing interfaces proves to be inefficient and inflexible where hands-free control may be preferable.

Proposed Solution

The proposed system utilizes computer vision and machine learning to enable natural, gesture-based drone operation, removing physical controllers entirely. By interpreting hand movements through a live camera feed from a device, the system will track and interpret hand gestures in real time, mapping them to flight commands such as take-off, landing, directional movement, and landing.

This creates a more intuitive, standardised, and accessible interaction model, while maintaining precise control. Safety-critical features will be incorporated, including signal-loss hovering and emergency override protection.

High-Level System Overview

At a high level, the system consists of six integrated layers:

1) Hardware input – Live camera input and drone telemetry data.
2) CV and gesture pipeline – Interpreting gestures and mapping them to drone commands. The core intelligence layer of the system
3) Backend API – The orchestration backbone of the system, enabling real time communication.
4) Frontend – The user facing monitoring and control dashboard, including live statistics and status monitoring.
5) Application deployment – Platform specific packaging and delivery, allowing compatibility on a variety of systems.
6) DevOps and Docs – The layer for deployment reliability, collaboration, and maintainability.

Proposed Technologies

The tech stack was chosen to maximise performance, taking into consideration community support, maintainability, and alignment with the project's computer-vision and drone control requirements. These proposals are not concrete, and in some cases, we provide multiple options, allowing some flexibility until we have more details regarding the project and your specific needs.

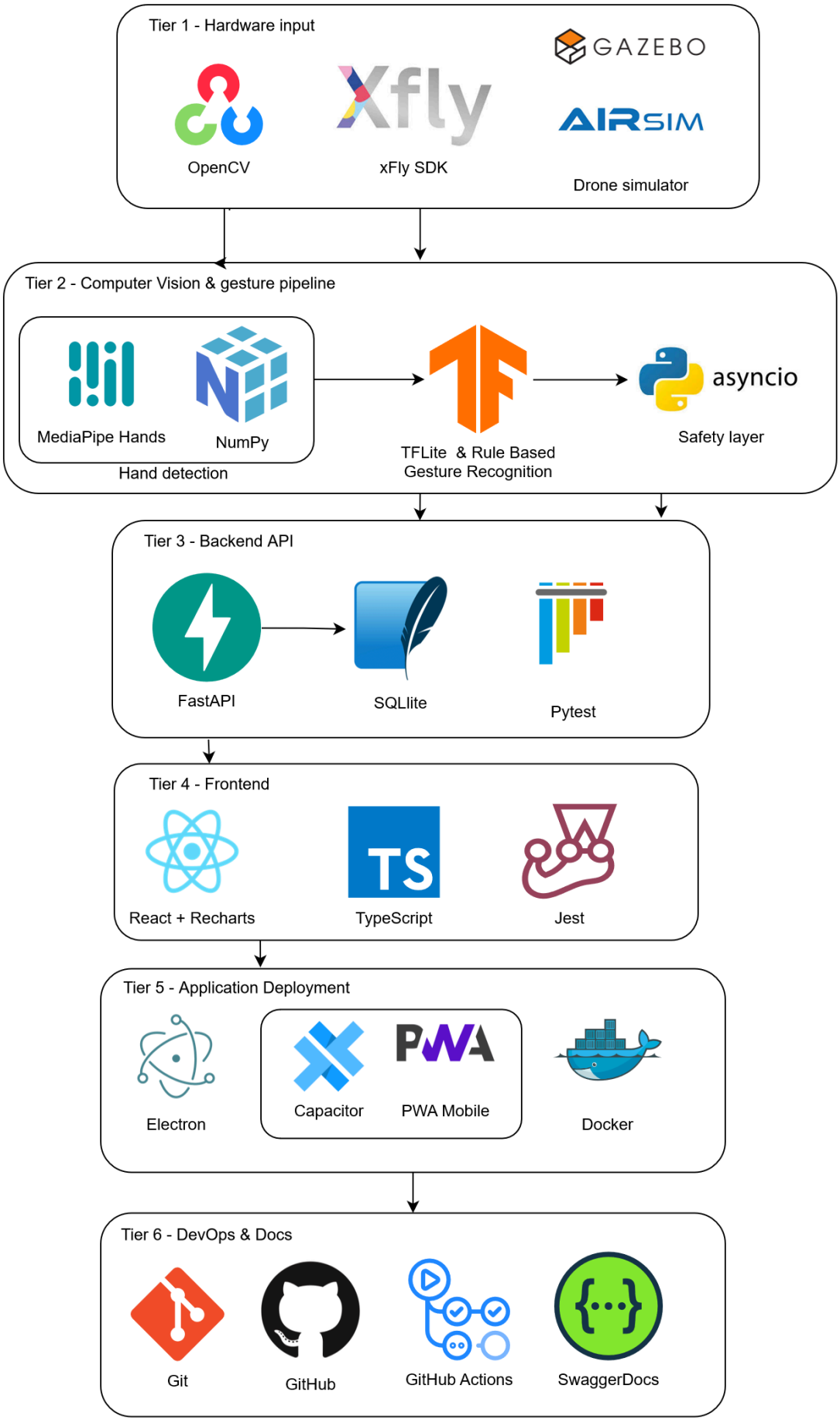

*Figure SEQ Figure * ARABIC 1: A high-level view of our proposed architecture*

Use case	Proposed Technologies	Version
CV & ML Backend	Python	3.11x
Video Capture & Processing	OpenCV	4.8+
Hand Detection	MediaPipe Hands	0.10+
Primary Gesture Recognition	Custom, Rule-based	N/a
ML Based Gesture Recognition	TensorFlow Lite	2.x
Numerical Processing	NumPy	1.26+
Real-Time Processing	Asyncio, bounded queue	N/a
Drone SDK	xFly SDK / DJI Tello SDK	Latest
Primary Drone Simulation	AirSim	Latest
Alternative Drone Simulation	Gazebo + ArduPilot	4..x
Backend API	FastAPI	0.110+
API Architecture	REST + WebSockets	N/a
Real-Time Communication	WebSockets	4.x
Frontend Framework	React + TypeScript	18/5.x
Data Visualisation	Recharts	2.x
PC Application Packaging	Electron	Latest
Mobile Packaging	PWA & Capacitor	Latest
Local Data Storage	SQLite	3.x
Alternative Database	PostgreSQL	15+
Backend Testing	Pytest	Latest

Frontend Testing	Jest	Latest
Documentation	SwaggerDocs, Overleaf	N/a
Containerisation	Docker	N/a
Version Control	Git + Github	N/a
CI/CD	Github Actions	N/a

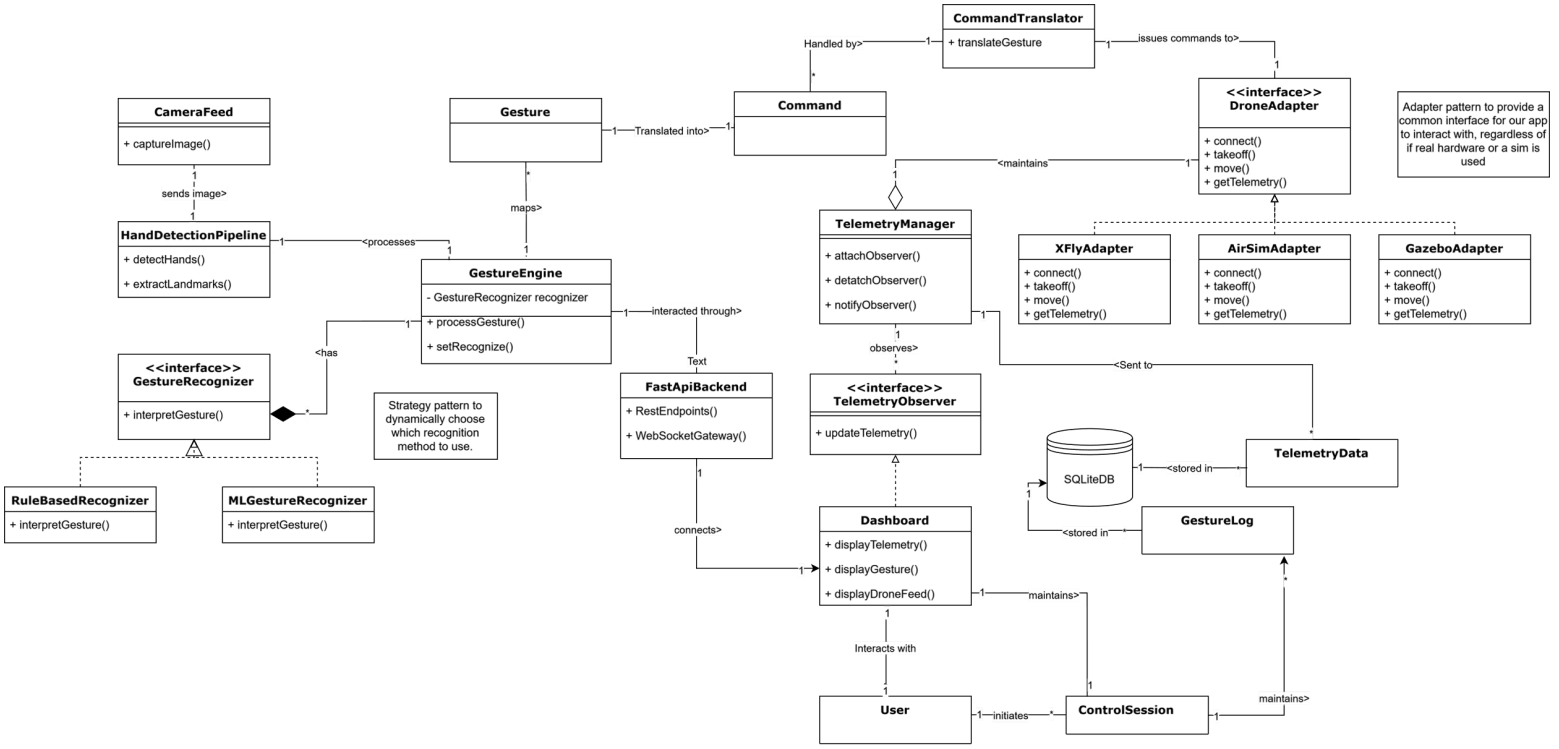

*Figure SEQ Figure * ARABIC 2: Our proposed domain model*

Proposed Technology Deep-dive

Python has been chosen as the primary language for the computer vision, machine learning, and drone integration pipeline. This is motivated by Python's establishment as the industry standard in this field, supported by its extensive ecosystem for ML and computer vision (OpenCV, MediaPipe, TensorFlow, etc.). It features easy integration with most drone SDKs, especially the XFly SDK, which is currently our preferred hardware platform

In terms of video capture and hand tracking, we propose OpenCV and MediaPipe Hands in tan dem. OpenCV will manage the live camera feed and image preprocessing, while MediaPipe Hands provides real-time hand landmark detection and tracking. They have the advantage of being CPU bound and efficient by default, making them ideal for lower power machines, and provide an ample suite of tools out of the box, including a suite of tools for image transformation and accurate tracking through pre-trained models

.

Gesture recognition forms the core functionality of our project. For this reason, we propose two complimentary approaches to this feature. Rule-based recognition, in which we maintain a set of mappi

ngs for positions to specific actions, based on data received from our hand model (for instance, if the data points for the thumb are ‘pointed up’, raise the drone). This approach is deterministic, with minimal overhead and latency.

However, as a secondary or future enhancement TensorFlow Lite is proposed for machine learning based gesture recognition. This approach is suitable for handling more complex or ambiguous gestures, and remains lightweight enough for real time inference on CPU based systems. Importantly, it can be introduced later on without requiring major architectural changes to the existing pipeline.

For numerical processing, we have opted for NumPy with real-time calculations in mind. The seamless integration with OpenCV and MediaPipe is certainly helpful, especially with the need for handling arrays, distances, angles, and vector mathematics for gesture identification.

Given the requirement for real-time actions on portable/low-end hardware, concurrency would certainly be preferred. We propose the use of Asyncio and a bounded queue, to serve as a non-blocking pipeline for processing gestures, capturing frames, and issuing commands. This ensures low latency, and avoids a backlog of unprocessed frames, in the case that CPU time is limited.

The drone itself forms a central component of our project, and largely dictates the direction that it goes in. We propose the use of the xFly SDK, as suggested in the project proposal, or the DJI Tello SDK. Both of these are highly compatible with Python, and compatible drones fall within our budget.

Before working with physical hardware, we need to perform our initial development and extensive testing in a controlled, virtual environment. Two candidate platforms have been proposed, AirSim, which is more lightweight and should run with less resources, as well as Gazebo + ArduPilot, which offers more realistic physics at a higher performance cost. The final choice can be determined depending on the host machine and the degree to which we desire realism.

FastAPI has been selected for the backend framework for its use of the ASGI standard, allowing support for both REST and WebSockets features. This functionality is essential in our use case, as REST is best for our configuration, logs, and system status, whereas WebSockets will be employed for real time communication, perfect for gesture streaming and telemetry data through a persistent connection.

allows us the most flexibility, maintainability, and compatibility across platforms. React provides a scalable component-based architecture, while TypeScript improves reliability through static typing.

For native deployment on Windows, Linux, and macOS, Electron will be used to package the web dashboard as a native executable application without requiring a separate desktop codebase.

To support iOS and Android with minimal development overhead, the same React frontend will also be adapted into a Progressive Web App (PWA) with optional Capacitor wrapping if mobile packaging becomes necessary. This unified strategy minimizes duplicated error while maximising platform coverage.

Data storage needs to be fast and scalable, whilst still maintaining support for complex operations. For this, we propose SQLite to store gesture logs for playback, telemetry data, and commands. As an alternative for larger scale or long-term storage requirements, PostgreSQL is included as an alternative, for its stronger scalability and more advanced capabilities if system complexity increases over time.

Backend testing will be conducted using Pytest, which integrates effectively with FastAPI and related frameworks. Pytest provides a flexible and readable structure for unit, integration, and API endpoint testing, ensuring backend reliability throughout development.

Frontend testing will be handled using Jest, as the team is already experienced with its use in JavaScript and TypeScript environments. Jest is widely adopted for

testing React applications and offers strong support for mocking, component testing, and automated validation of frontend logic.

Documentation will be handled primarily through SwaggerDocs and Overleaf. Swagger integrates directly with FastAPI to generate live, interactive API documentation automatically. Overleaf is chosen for formal written documentation due to its collaborative LaTeX support and professional formatting capabilities.

Docker has been selected for containerisation to ensure that the application runs consistently across different development and deployment environments. This reduces hardware and environment specific issues and simplifies setup, making collaboration significantly more reliable.

Github will serve as the project's version control platform, as it remains the industry standard software for collaborative software development. We intend to use a modified Git flow strategy resembling trunk-based development, consisting of main, development, and feature branches.

GitHub Actions will be used to automate continuous integration and deployment workflows. This ensures that all code merged into shared branches is automatically tested, linted, and checked against quality standards. This improves code reliability and reduces the risk of unstable builds reaching production environments.

Why Choose Codex Merchants?

Ayush Beekum

He has delivered production web applications for real clients using React and Python, demonstrating equal strength on both frontend and backend development.

He has completed several standalone projects from concept to deployment, giving him the self-sufficiency to own features end-to-end, one of them being a python coded system that fetches real-time data from an API and reflects in real-time on the application.

His language proficiency includes Python, JavaScript, TypeScript, SQL, C++, Java, and PHP, meaning he can work across our entire proposed stack without struggling to understand any of the frameworks and languages from the proposed tech stack. He is also very consistent in integration and bug fix identification and solution as on every major project he has contributed greatly to this aspect.

Top Languages/Frameworks:

1) Java
2) C++
3) SQL
4) Python
5) Typescript

https://github.com/Ayush-B99

www.linkedin.com/in/ayush-beekum

Shavir Vallabh

Shavir has a strong background in AI and optimization. He has implemented and worked with machine learning and heuristic optimization techniques, including recurrent neural networks (RNNs) in PyTorch, the Genetic Algorithm, and the Quantum Approximate Optimization Algorithm (QAOA), applying these methods in cybersecurity contexts such as hashed password cracking and search-space optimization.

He also holds an interest in high-performance computing (HPC), low-level optimization, and systems architecture. He led the Tuks team to the finals of the CHPC Student Cluster Competition, involving DevOps orchestration, compiler tuning, benchmarking, and performance profiling under strict resource constraints. This experience strengthened his expertise in Linux systems, parallel workloads, cluster deployment, and optimization at both software and hardware interaction layers.

Currently, Shavir works part-time as a TechTeam member in the University of Pretoria's Computer Science department, where he contributes to both systems administration and software engineering tasks, including maintaining Linux-based infrastructure, and developing full-stack internal tools.

Top Languages/Frameworks:

1) Python
2) C
3) React + Node.js
4) C#
5) Java

 https://github.com/ShavirV

 https://za.linkedin.com/in/shavir-vallabh

Diya Narotam

She has hands-on experience working with Python and NumPy for data manipulation.

She actively applies OOP principles using C++ and Java and is proficient in SQL for database querying and management. She participated in a hackathon focused on a school website update competition, where her team secured 1st place.

In addition to development, she makes contributions to UI/UX design and frontend development. She has also worked with Progressive Web Apps (PWAs) and gained practical experience in building offline-capable mobile applications.

To ensure code reliability and quality, she has used multiple testing libraries including Playwright, Vitest, and Jest, giving her a solid foundation in both unit and end-to-end testing.

Top Languages/Frameworks:

1) Java
2) C++
3) JavaScript
4) Python
5) Typescript

 https://github.com/deexglitch

Chinmayi Santhosh

She has considerable experience in statistical and data analysis skills.

She is well versed in UI engineering and working with several different UI frameworks such as React and Tailwind CSS.

She has experience writing unit and integration testing in TypeScript to ensure frontend reliability and catch regressions early.

Across her university projects, she has made significant contributions to frontend development, delivering clean, functional interfaces and collaborating effectively with the back-end team members.

Top Languages/Frameworks:

1) Java
2) C++
3) JavaScript / TypeScript
4) Python
5) React

 https://github.com/ChinmayiSanthosh

Jaitin Moodally

He has experience working over several different operating systems and in several different environments.

He is greatly experienced through homelabbing in running different applications on Virtual Machines with low specifications.

He is also experienced in hosting multi-user applications and low-level programming.

Top Languages/Frameworks:

1) Java
2) C++
3) C
4) Python
5) Typescript

 https://github.com/Wave2055

Software Engineering Methodology

We intend on using Agile as our choice of software methodology. We will be utilizing scrums, involving weekly stand-up meetings, Sprint planning and Sprint review. We will make use of additional meetings when necessary and will only involve the necessary team members and will include a meeting summary for team members that were not present.

Feature Driven development will form a cornerstone for helping to design and assign tasks and goals within the agile methodology.

The possibility of needing to learn new technologies during the duration of the project is very likely. We intend to use a On The Job style of learning in order to allow team members to learn new skills and frameworks while working towards a goal.

Current Shortcomings

Our team is less experienced in the use of CI/CD. To remedy this we intend to work with our industry partner to setup our CI/CD pipeline to run on the correct branches and to use the appropriate testing strategy for this project.

Team Strengths

Our team has an outstanding track record. Across every group project in our degree, the five of us have consistently scored above 90%, demonstrating a standard of excellence and an ability to work together that most teams cannot claim.

We are quite comfortable with the proposed tech stack; our team has a diverse skillset that covers every base in terms of the software engineering pipeline.

This project holds a special interest to our group, the opportunity to work on such a unique project, interacting with real hardware in such a dynamic way is a rare and

exciting prospect that we are enthusiastic to experience and lend our unique skillset towards.