CyDrone

Team sdmay19-35

Dr. Ali Jannesari, Client & Adviser

Bansho Fukuo, Ian Gottshall, Jianyi Li, Jawad M Rahman, Sammy Sherman & Mehul Shinde.

Email: sdmay19-35@iastate.edu
Website: https://sdmay19-35.sd.ece.iastate.edu

Problem Statement

What does the client have?

➔ A drone with a camera

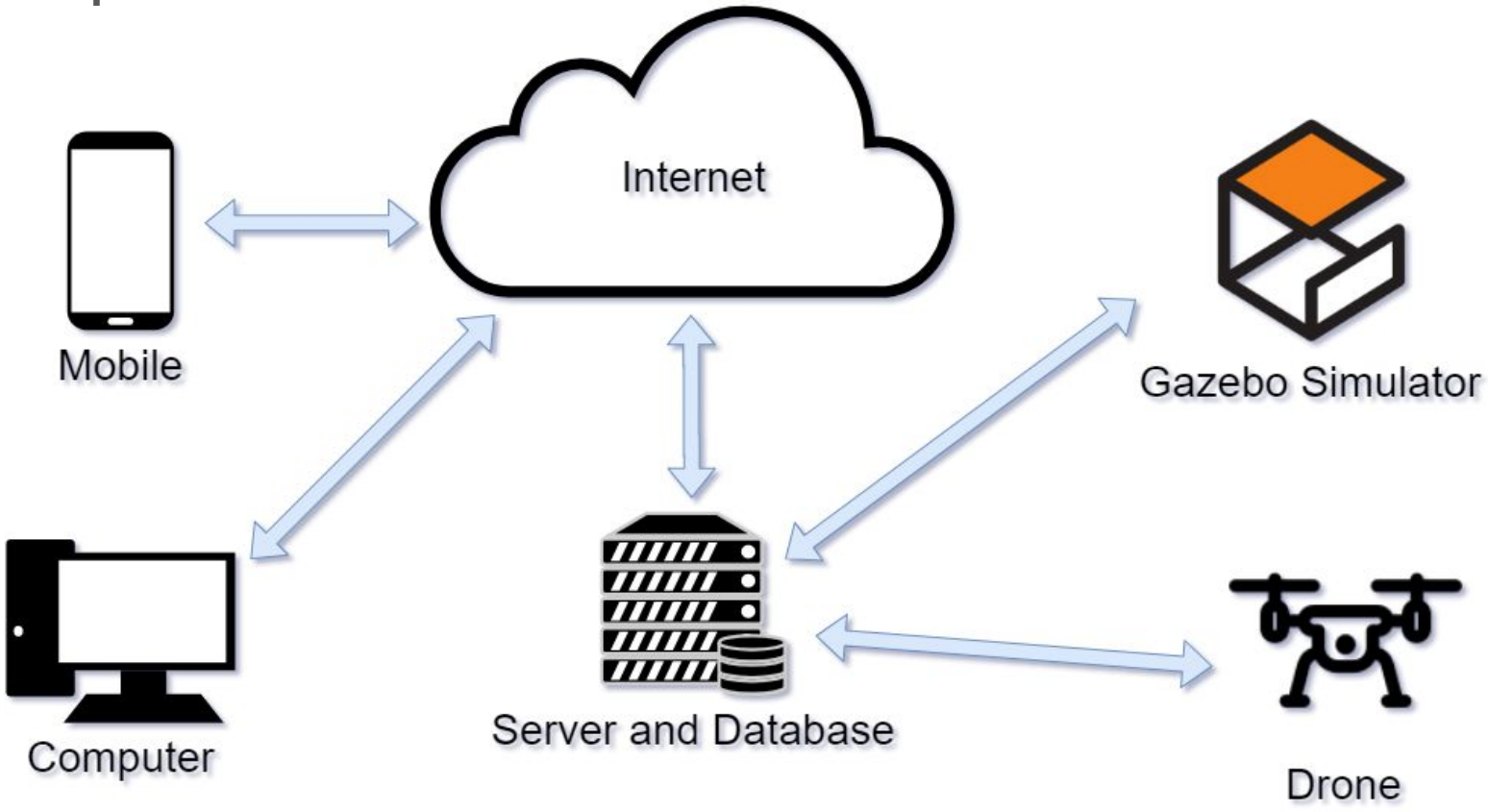
What does the client want?

➔ A cross platform and open-source application to simulate and control the drone and digitally recreate the real flight environment inside the simulator

Our solution:

➔ CyDrone - an open-source web-application built using ROS, ReactJS, Gazebo and WebODM

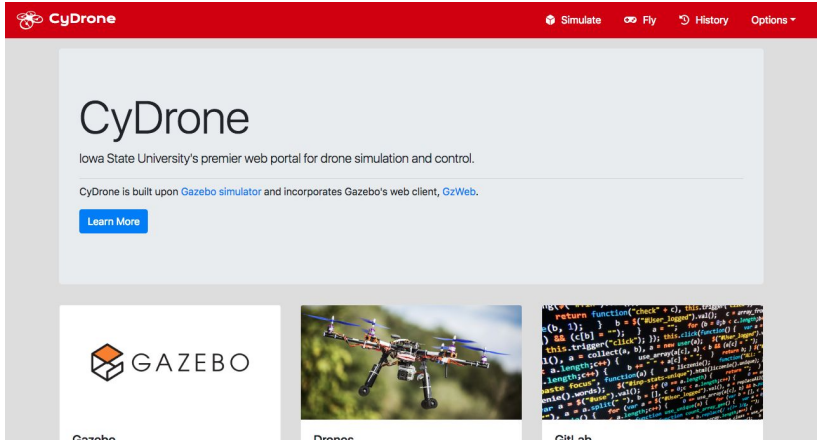
Conceptual Sketch

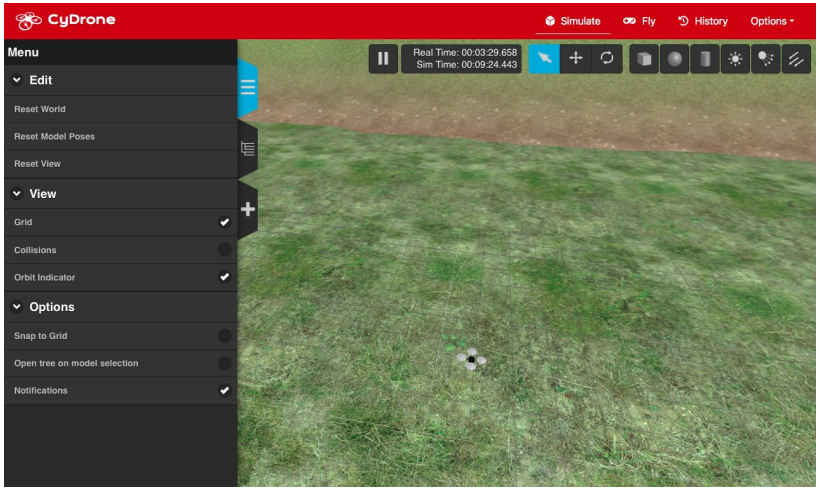


User Interface Description

A ReactJS based web application consisting of 4 main components:

- The homepage displaying news, tutorials and other useful information
- A page for allowing the user to run and control a drone simulation
- A page for launching and controlling a real drone
- A page for viewing flight history and statistics





Functional Requirements

Drone Simulation

The web application should allow the user to control a drone in a simulation from their web browser of choice.

Drone Control

The user must be able to control a real drone using their keyboard, onscreen controls, or a predetermined route.

Computer Vision Generated Environments

Environments used in the simulator shall be generated by applying computer vision techniques to the video feed captured from the drone’s onboard camera.

Non-Functional Requirements

Safety

When the connection to the drone is lost or interrupted, a collision occurs, a fault occurs with the drone, or any other hazardous event occurs, the situation must be handled safely and the user must be notified.

Scalability

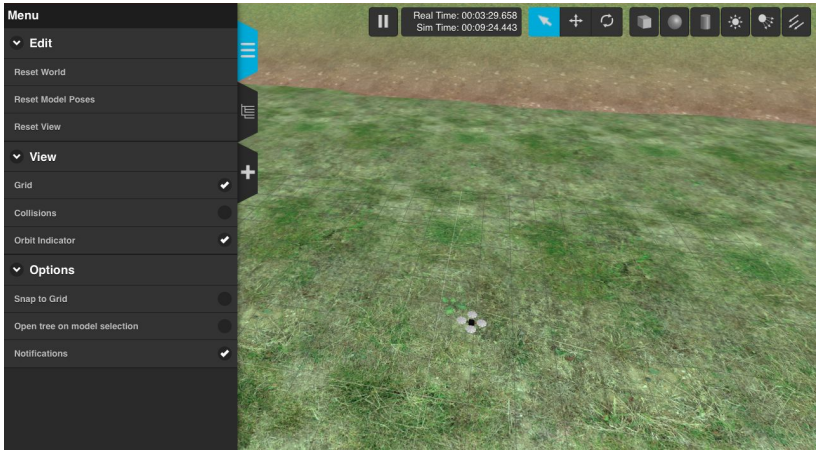
The service must be able to handle a growing number of active users without allowing performance to suffer.

Compatibility

Any modern browser, including those on mobile devices, must be able to access and interact with the web-portal and all of its features.

Market and Literature Survey

- Gazebo
  - Open source robot simulator
  - Customizable environment
  - Supports C++ plugins
  - Supports ROS integration
  - Drawbacks:
    - Resource-heavy
    - Development is not cross-platform
- AirSim
  - Open source autonomous vehicle and drone simulator by Microsoft
  - Built in Unreal Engine
  - Excellent graphics
  - Drawbacks:
    - Need to learn Unreal Engine
    - No ROS integration



Source: “Aerial Informatics and Robotics Platform.” *Microsoft*, https://www.microsoft.com/en-us/research/project/aerial-informatics-robotics-platform/.

Deliverables

1. Web application that will be able to simulate a drone in a variety of environments, be able to control the drone and provide necessary data in the process
2. Documentation:
    a. Code
    b. Contribution of the members
3. Manual:
    a. Interaction of the front-end and back-end
    b. Component diagrams and flowcharts

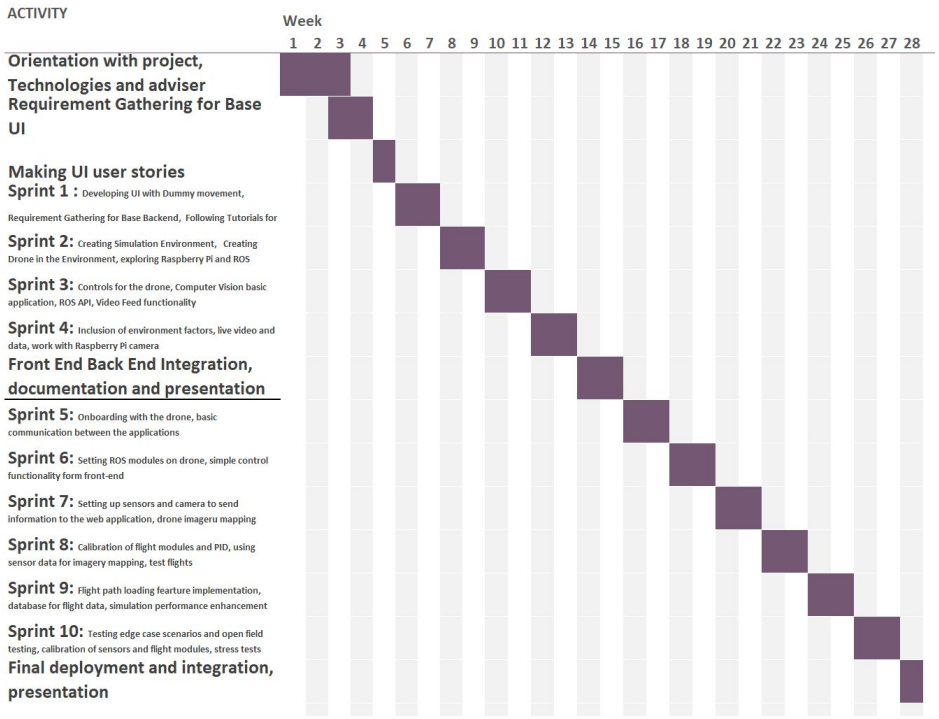
Work Plan

- Work Distribution
  - Sammy Sherman - Front-End Developer & Report Manager
  - Ian Gottshall - Front-End Developer & Scrum Master
  - Bansho Fukuo - Back-End Developer & Test Engineer
  - Jianyi Li - Back-End developer & Test Engineer
  - Mehul Shinde - Back-End developer & Project Lead
  - Jawad M Rahman - Back-End developer & Meeting Manager
- Resource Requirements
  - DJI Matrice 100 drone - $3299
  - Project duration - 28 weeks, 9 hours per week
  - Server running Ubuntu with at least 30GB storage and ample RAM
  - Camera
  - Sensors if required

Source: “Buy Matrice 100.” *DJI Store*, store.dji.com/product/matrice-100.

Work Plan

- Project Schedule
  - 10 sprints in total
  - 4 sprints this semester
    - Developed UI, gathered resources for the back-end
    - Created simulation environment, created drone in the environment, researched on Raspberry Pi and ROS
    - Created controls for the drone, Setup ROS, computer vision, video feed
    - Inclusion of environment factors, live video and Raspberry Pi camera

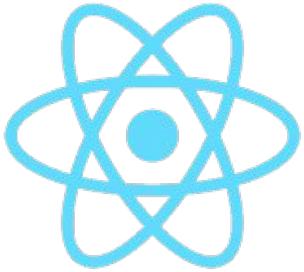


Work Plan

- Risk
  - Gazebo is resource heavy
  - Calibrating the drone with the simulation
  - Incompatible software
  - General unfamiliarity with technology used
- Mitigation
  - Working with the client for equipment
  - Conducting thorough research and testing often

System Analysis

- ReactJS
  - Prior experience
  - React Native
  - Node support
  - Other options: AngularJS
- Gazebo
  - Open-source
  - Developer community
  - GzWeb
  - Other options: AirSim



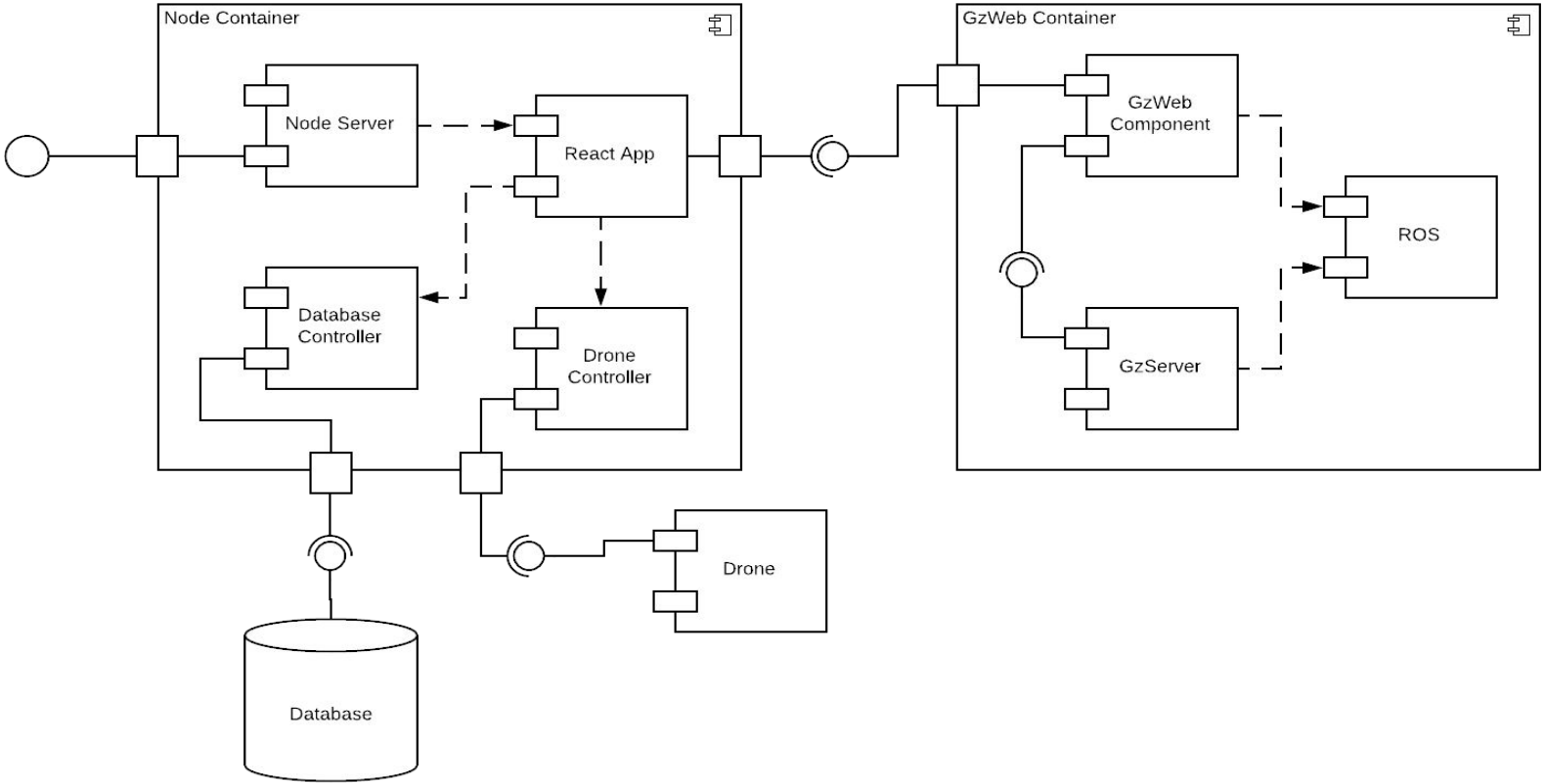
React

GAZEBO

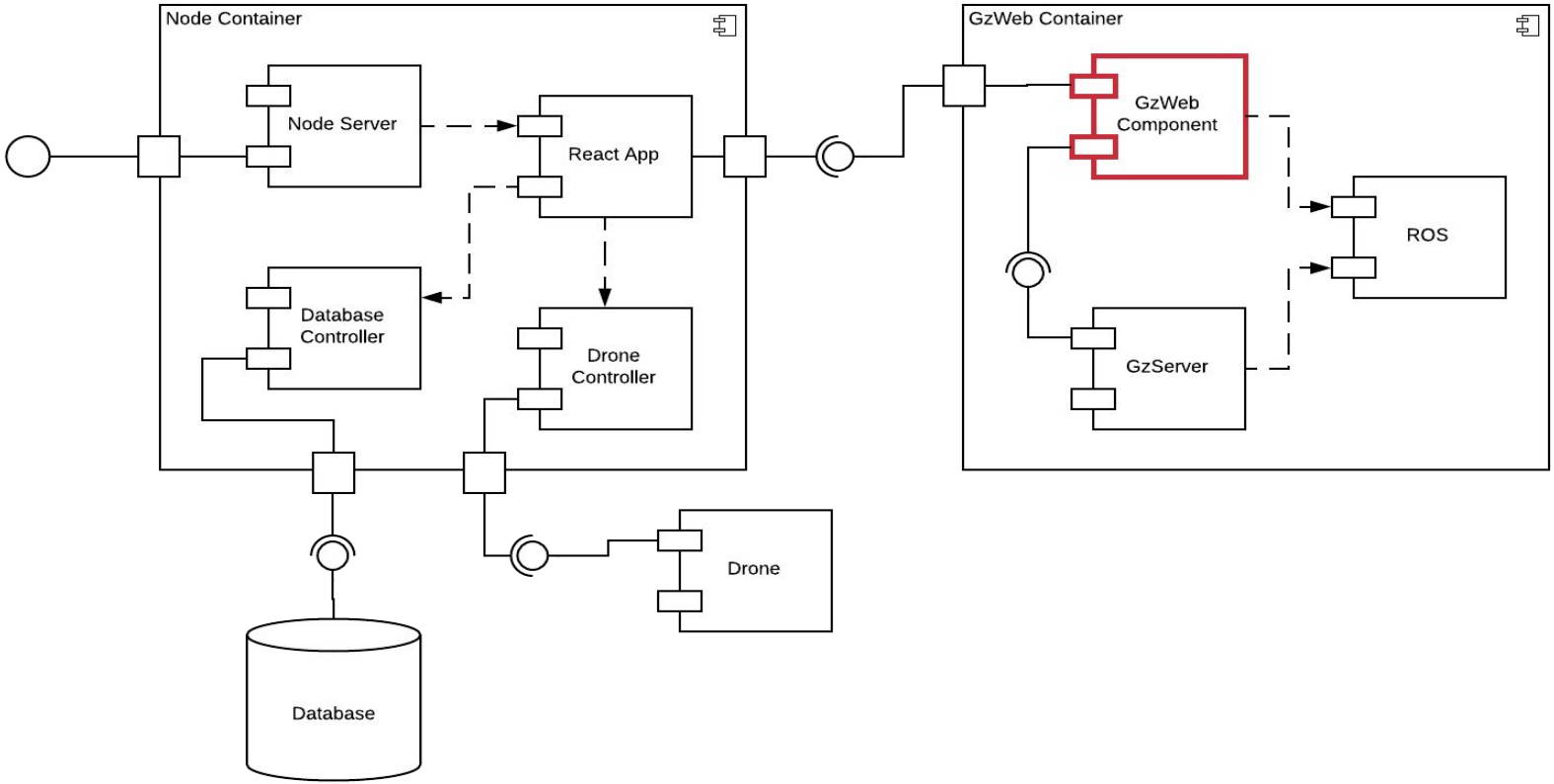
System Analysis

- ROS
  - Client's system support
  - Modules for flight movement
- WebODM
  - Open-source
  - Documentation
  - Wide array of output types
  - Other options: DroneDeploy
- Backend languages
  - C
  - C++
  - Python: WebODM services

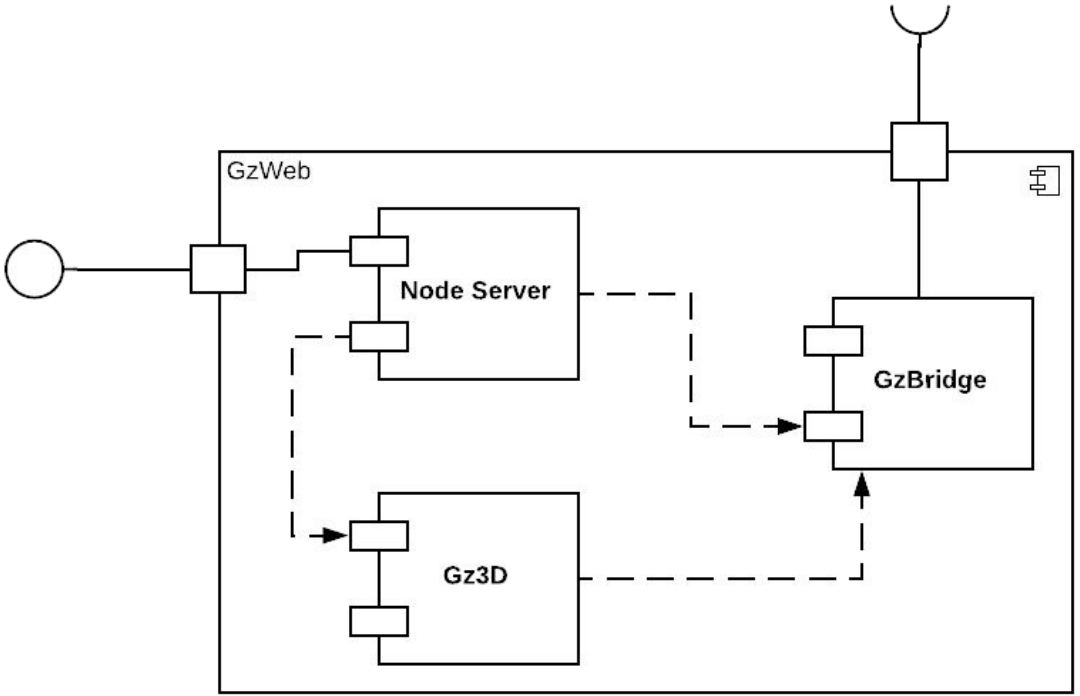
Detailed Design - System Architecture



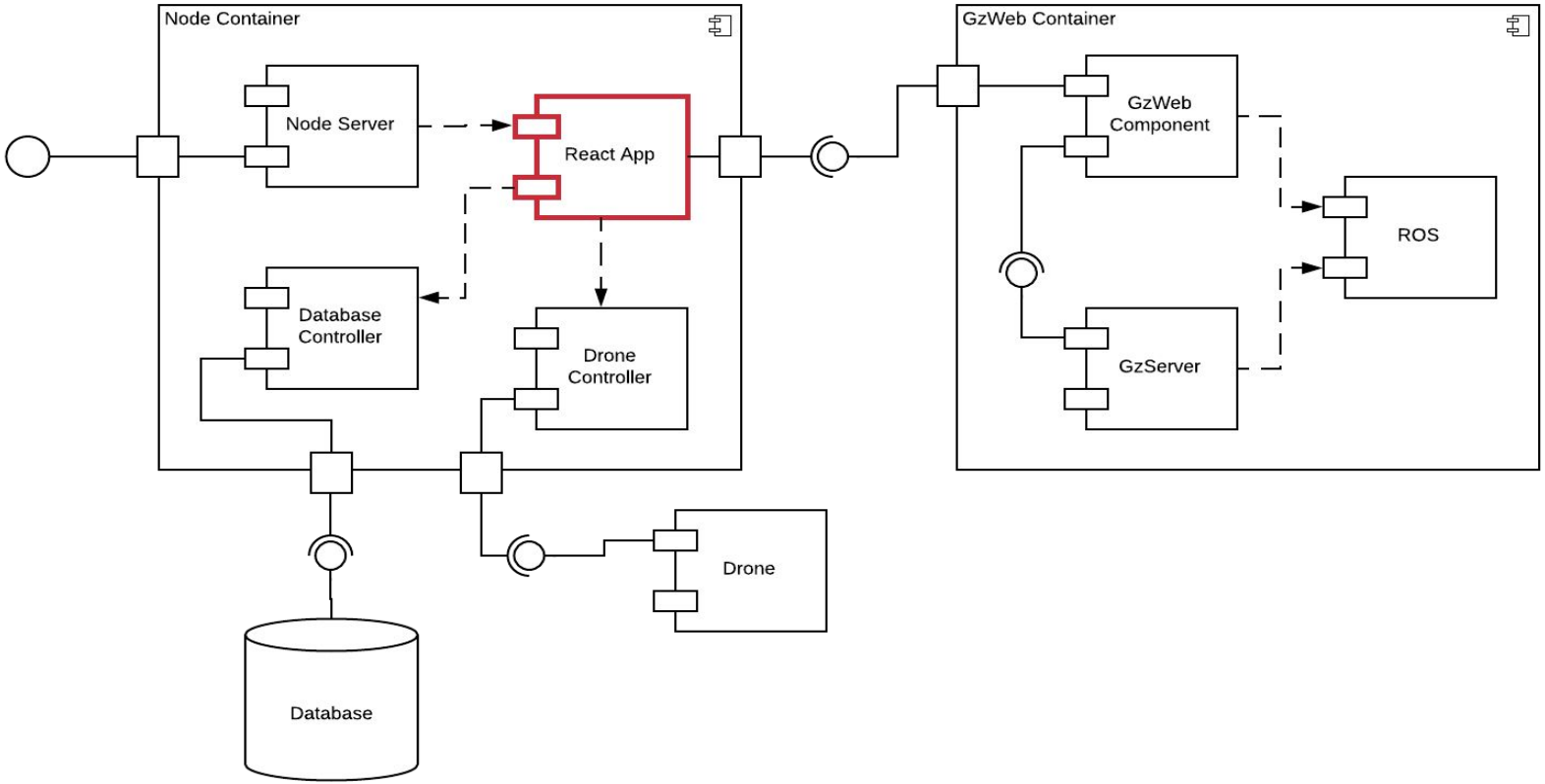
Detailed Design - System Architecture



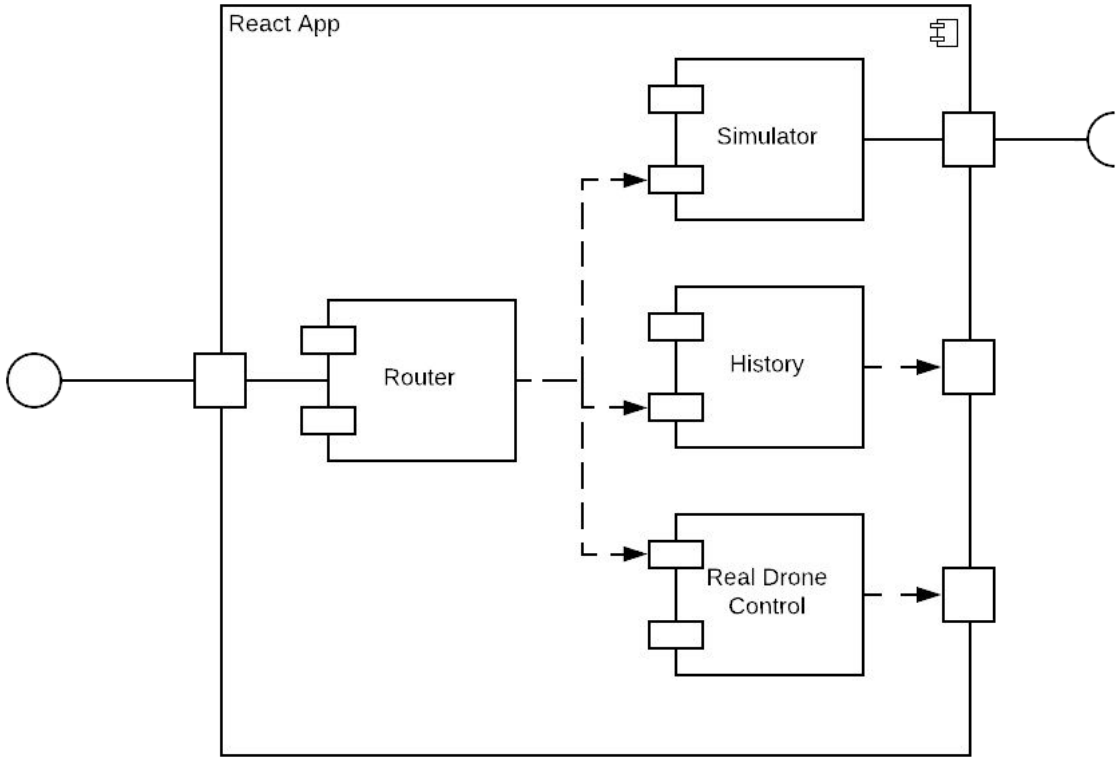
Detailed Design - System Architecture - GzWeb



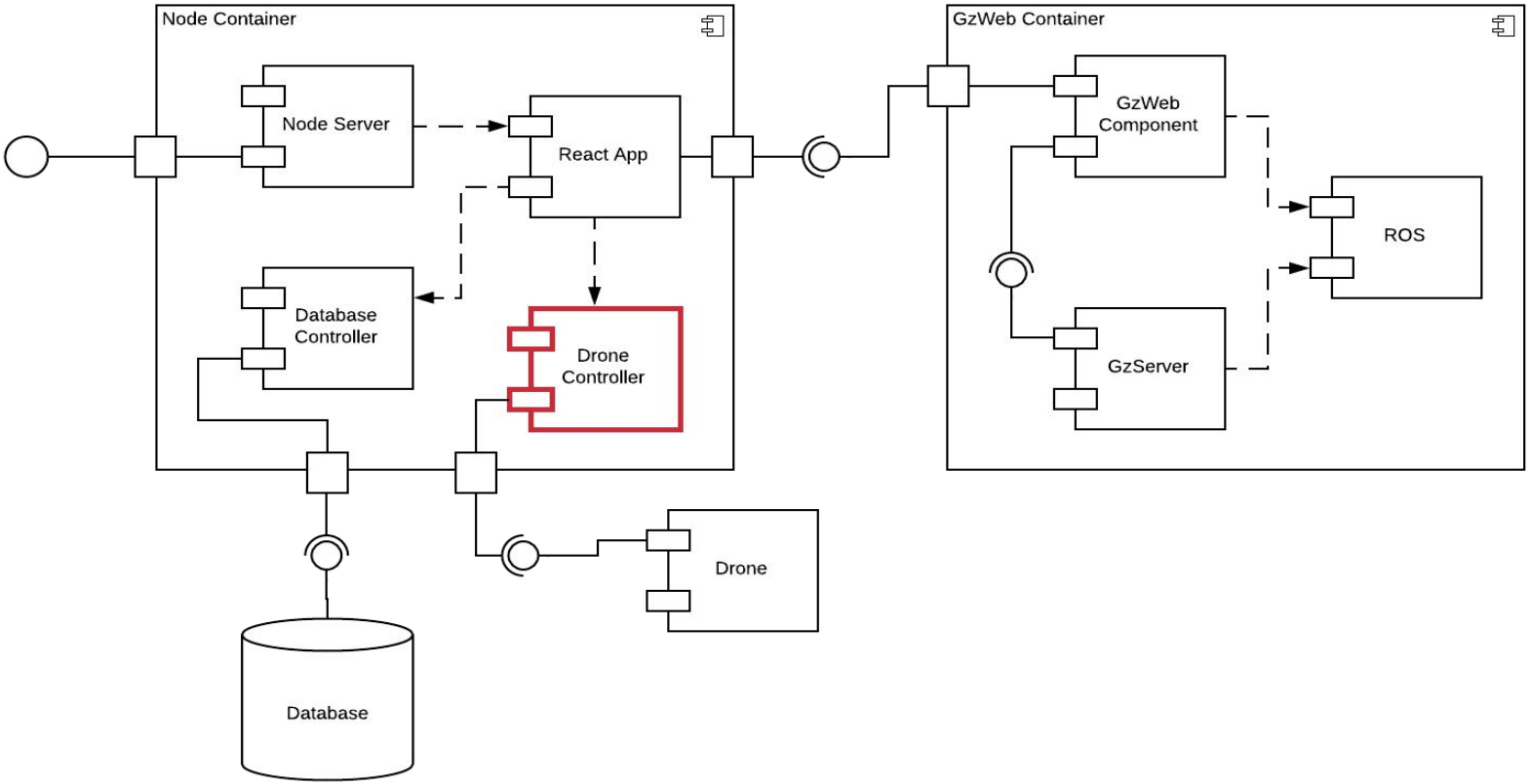
Detailed Design - System Architecture



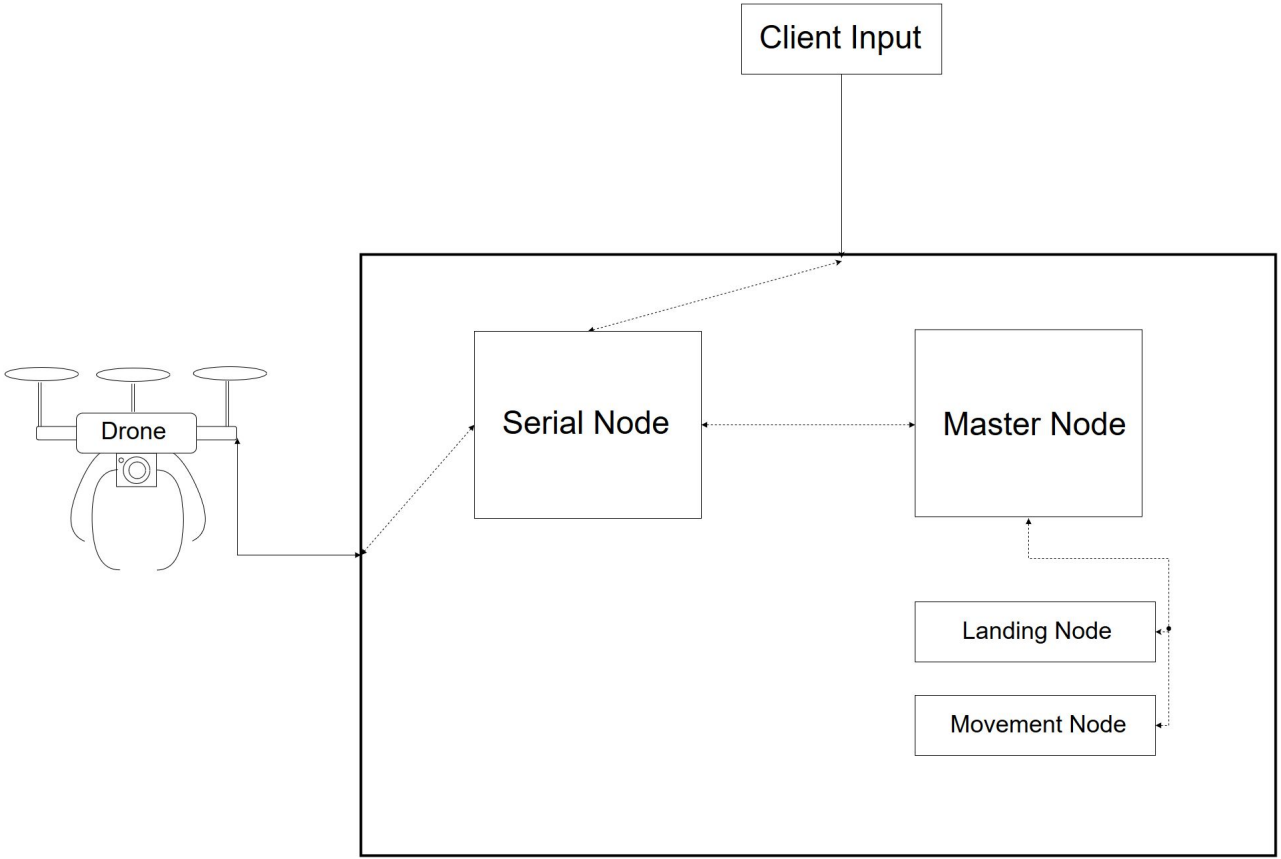
Detailed Design - System Architecture - React



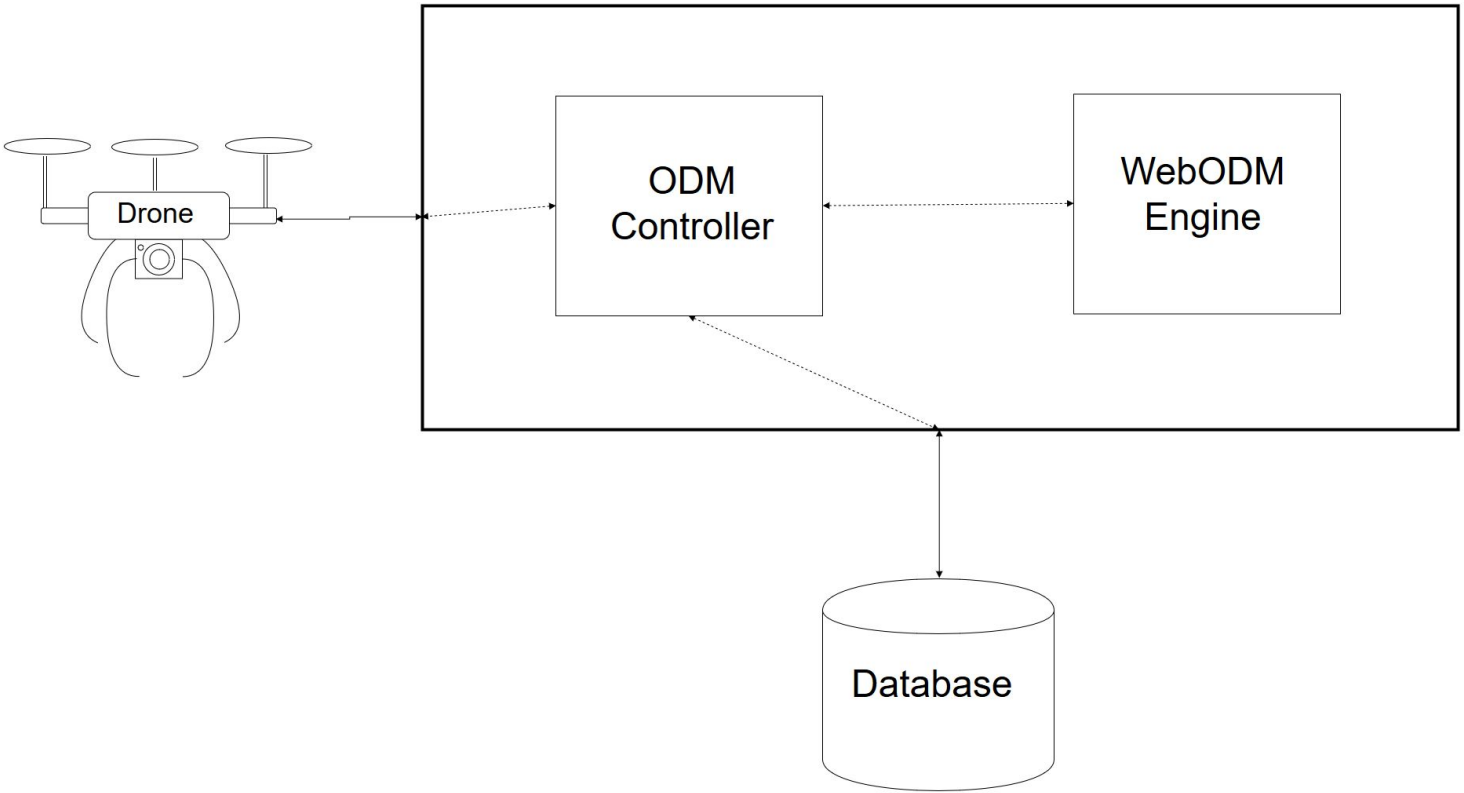
Detailed Design - System Architecture



Detailed Design - System Architecture - ROS



Detailed Design - System Architecture - WebODM



Test Plan: Simulation Functionalities

Login to the simulation site with a test user account, and load drone simulation environment.

- Controls over the UI control panel
- Controls over the Keyboard controls
- Entering valid command into the terminal.

Success Criteria

Each command responds in less than 0.25 seconds and performs the correct action.

Failure Criteria

Any result other than the success criteria.

Source: “Velocidrone FPV Simulator vs. Reality - NCAR Racetrack” William Thielicke, www.youtube.com/watch?v=ewHdnTiNL3M.

Test Plan: Environment Editor Functionalities

Login to the simulation site with a test user account, and load a simulation environment for editing.

- Placing an object into the environment.
- Saving and reloading

Success Criteria:

Each command responds in less than 0.25 seconds and performs the correct action.

Failure Criteria:

Any result other than the success criteria.

Test Plan: Calibration

Login to the simulation site with a test user account, and load a simulation environment.

- Synchronize the simulation to the drone.
- Using each of the basic movement and rotation controls and verify that the positional data match after each trial.
- Repeat the previous step for improving accuracy

Success Criteria:

The change in position/rotation observed differs from the simulation by less than a 0.1% margin of error.

Failure Criteria:

Any result other than the success criteria.

Source:“Why use software for calibration management?” qedge, qedge.sarjen.com/why-use-software-for-calibration-management/.

Test Plan: Flight Tests

Login to the simulation site & load a Drone control.

- Synchronize the simulation with the drone & make the drone take off.
- Control the basic movements and rotational controls and ensure that the drone is doing exactly as asked and verify the positional data match.
- Testing at different altitudes or conditions.

Success Criteria:

The drone behaves as expected at different altitudes.

Failure Criteria:

Any result other than the success criteria.

Source: “DJI MATRICE 100 TEST FLIGHTS.” heliguy, www.heliguy.com/blog/2015/08/21/dji-matrice-100-test-flights/.

Test Plan: Video and Imaging Tests

Login to the simulation site & load a Drone control.

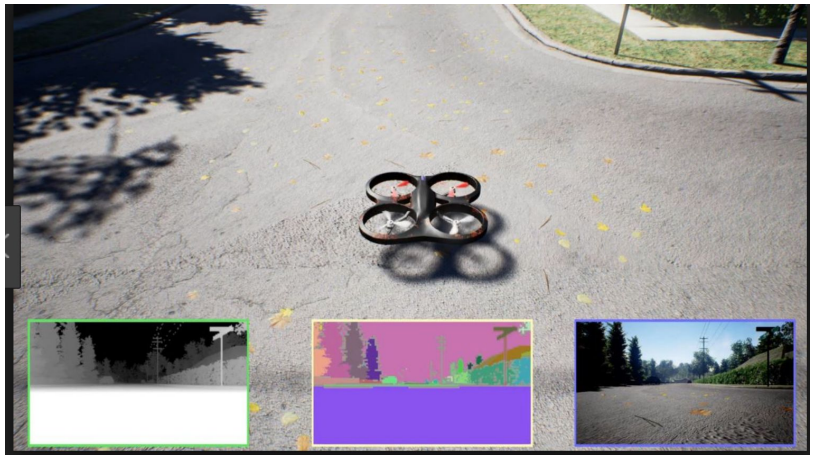
- Take off the Drone to certain attitude.
- Ensure that the simulation is receiving video from the drone’s camera.
- Ensure that the quality of this video is as expected and there is little to no lag in the video module of the simulation.

Success Criteria:

The drone delivers high quality images and videos.

Failure Criteria:

Any result other than the success criteria.



Source:”AirSim: A Simulator to Help AI Research for Use in Drones” expouav, www.expouav.com/news/latest/airsim-simulator-help-artificial-intelligence-research-use-drones/

Q&A

THANK YOU

Technology Platforms Used