IntelliGaze: A Wearable AI Camera System

Microprocessor Lab Project Presentation

Touhidul Alam Seyam (230240003) Eftakar Uddin (230240004)
Tasmim Akther Mim (230240025) Shafiul Azam Mahin (230240022)
Muntasir Rahman (230240002)

Department of Computer Science and Engineering
BGC Trust University Bangladesh

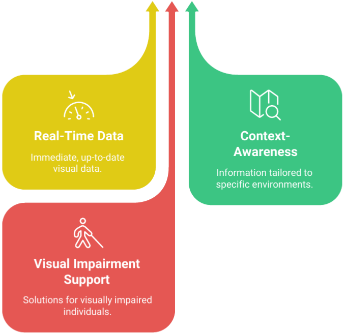
June 16, 2025

Outline

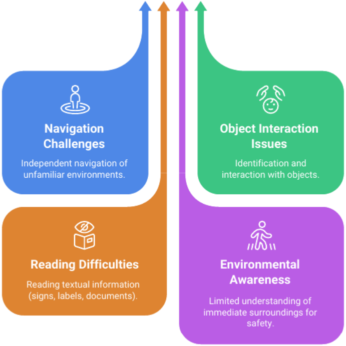
Introduction: The Need for Enhanced Vision

Addressing Visual Accessibility Challenges - Motivation

Core Problem



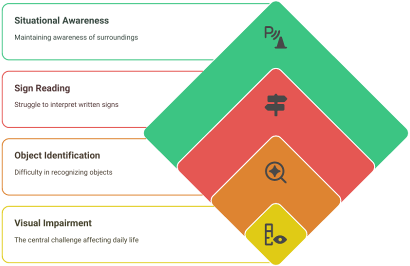
Limitations that can hinder



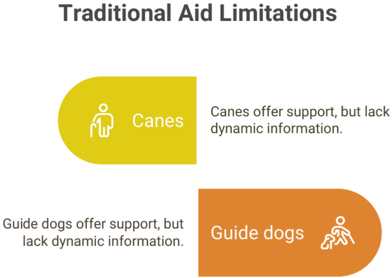
Problem Statement & Project Aim

Defining the Core Issue and Our Solution

Core Problem



Limitations that can hinder

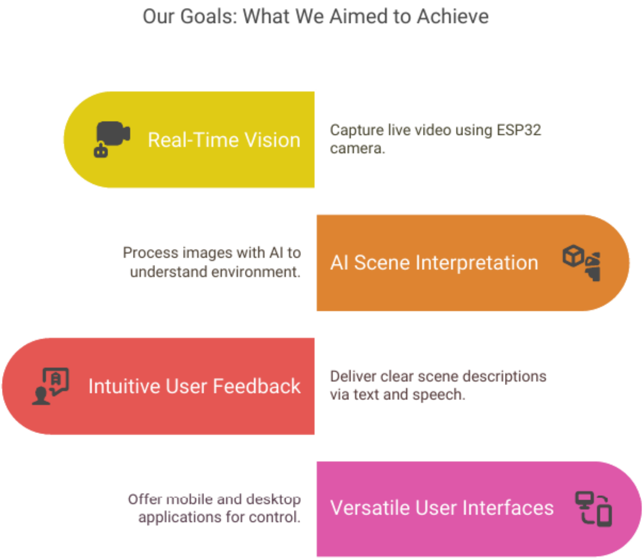


Our Aim with IntelliGaze

To provide a system that actively processes visual input from a wearable camera and delivers concise, relevant descriptions, thereby mitigating these challenges and empowering users.

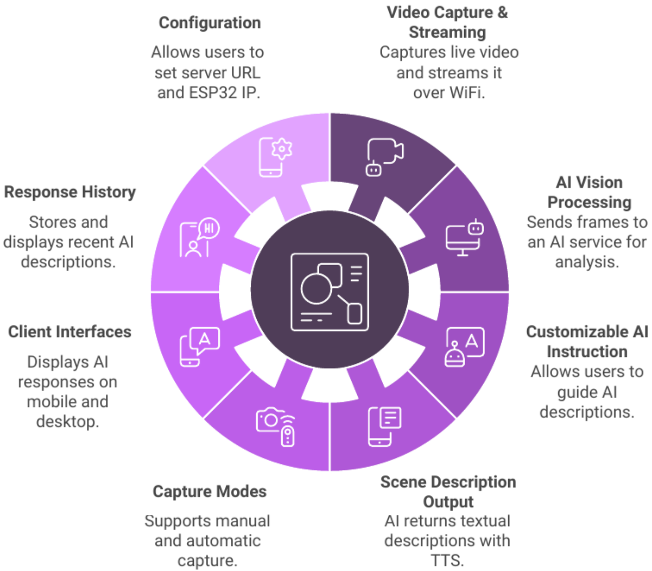
Project Objectives

Our Goals: What We Aimed to Achieve



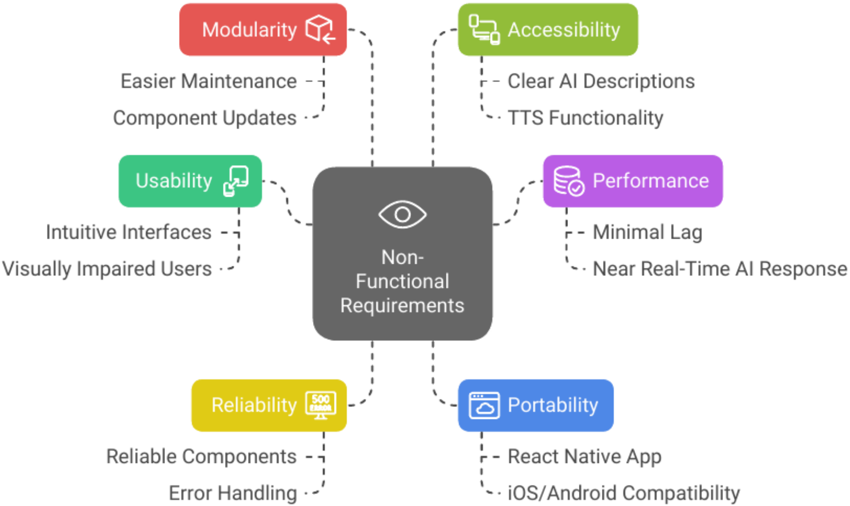
Functional Requirements

What the System Must Do



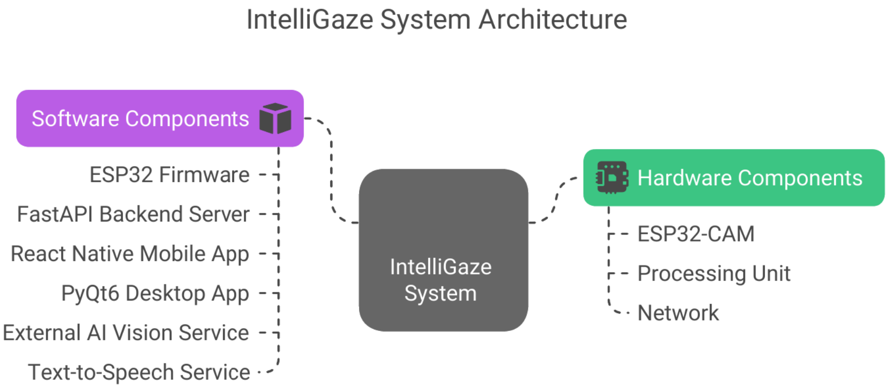
Non-Functional Requirements

Quality Attributes of the System



Hardware & Software Components

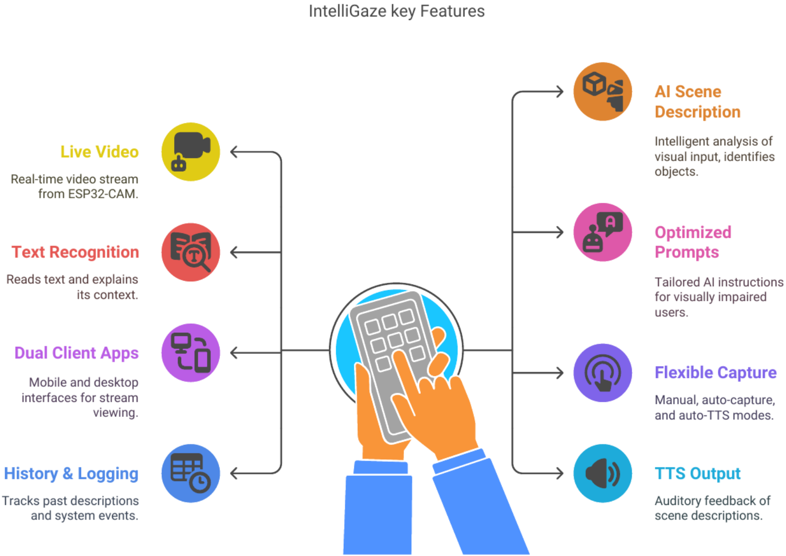
The Building Blocks of IntelliGaze



Primary & Secondary components of the IntelliGaze system.

Key Features

Core Capabilities of IntelliGaze



Detailed Circuit Diagram

ESP32-CAM to ESP32-CAM-MB Connections

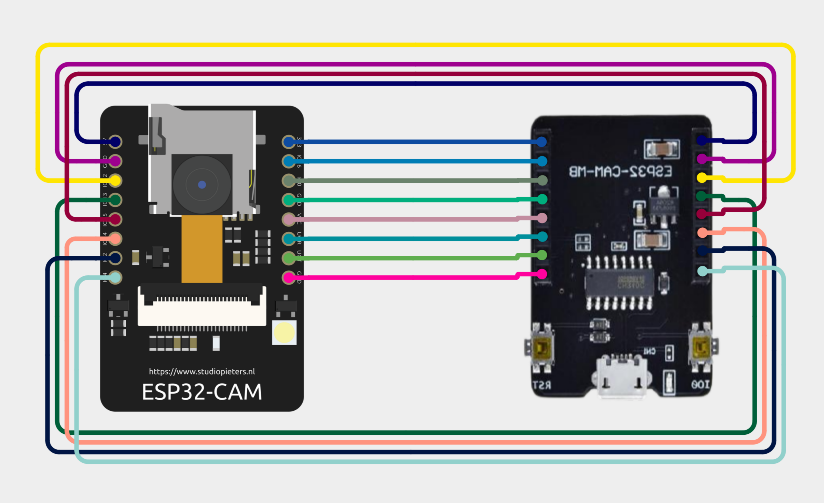


Figure: Essential pin connections for power, programming (via MB board), and camera operation.

Code: ESP32 Camera Configuration

`camera-feed.ino` - Initialization in `setup()`

```
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
// ... Pin definitions for D0-D7, XCLK, PCLK, VSYNC, HREF, SIOD,
    SIOC, PWDN, RESET ...
config.pin_d0 = Y2_GPIO_NUM; config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM; // ... and other pins
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

if(psramFound()){
  config.frame_size = FRAMESIZE_VGA; // 640x480
  config.jpeg_quality = 10; // Higher quality
  config.fb_count = 2;      // Use 2 frame buffers
} else {
  config.frame_size = FRAMESIZE_QQVGA; // Lower resolution
  config.jpeg_quality = 15;            // More compression
  config.fb_count = 1;
}
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) { /* Handle initialization error */ }
```

Listing 1: Camera Settings

Code: ESP32 MJPEG Streaming

camera-feed.ino - stream_handler Function

```
static esp_err_t stream_handler(httpd_req_t *req){
  while(true){
    fb = esp_camera_fb_get(); // Get frame from camera
    if (!fb) { /* Error handling */ res = ESP_FAIL; }
    else {  res = httpd_resp_send_chunk(req, "--frame", strlen("--frame"));
      if(res == ESP_OK){
        size_t hlen = snprintf(part_buf, 64,
          "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n", fb->len);
        res = httpd_resp_send_chunk(req, part_buf, hlen); }
      if(res == ESP_OK){  res = httpd_resp_send_chunk(req, (const char *)fb->buf, fb->len);}
      esp_camera_fb_return(fb); // Return frame buffer
    }
    if(res != ESP_OK){ break; }
    delay(30);
  }
  return res;
}
```

Listing 2: MJPEG Frame Send Loop

Code: FastAPI Vision Endpoint

Flask_server/server.py - AI Processing Request

```python
@app.post("/vision")
async def vision_endpoint(instruction: str = Form(OPTIMIZED_PROMPT)):
    global latest_frame # Obtained from background ESP32 stream worker
    image_b64 = base64.b64encode(latest_frame).decode("utf-8")
    image_data_url = f"data:image/jpeg;base64,{image_b64}"
    payload = {
        "max_tokens": 100,
        "messages": [{
            "role": "user", "content": [
                {"type": "text", "text": instruction},
                {"type": "image_url", "image_url": {"url": image_data_url}} ] }] }
    headers = {"Content-Type": "application/json"}
    async with httpx.AsyncClient(timeout=30) as client:
        resp = await client.post(AI_BACKEND_URL, json=payload, headers=headers)
        return {"response": data["choices"][0]["message"]["content"]}
```

Listing 3: Sending Frame to AI

Code: React Native TTS Utility

`inteligaze/utils/playGroqTTS.ts` - Audio Feedback

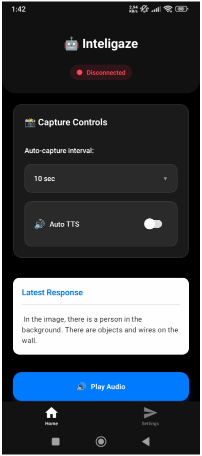
```
export async function playGroqTTS(text: string) {
  try {
    const response = await axios.post(
      'https://api.groq.com/openai/v1/audio/speech',
      { model: 'playai-tts', voice: 'Fritz-PlayAI',
        input: text, response_format: 'wav' },
      { headers: { 'Authorization': `Bearer ${GROQ_API_KEY}`},
        responseType: 'arraybuffer'}
    );
    const fileUri = FileSystem.cacheDirectory + 'tts.wav';
    const base64Audio = arrayBufferToBase64(response.data); // Helper
    await FileSystem.writeAsStringAsync(fileUri, base64Audio,
                                        { encoding: 'Base64' });
    const { sound } = await Audio.Sound.createAsync({ uri: fileUri });
    await sound.playAsync();
  } catch (error) { console.error('Groq TTS error:', error); }
}
```

Listing 4: Calling Groq TTS API

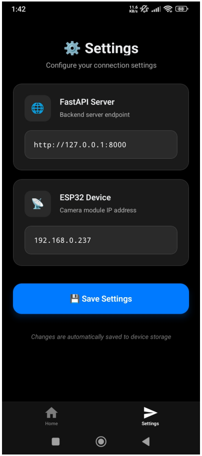
Demonstration: Mobile Application

IntelliGaze in Action - React Native App

Main Vision Interface



Settings Configuration



Key features shown: Connection status, capture controls (auto-capture interval, auto TTS), latest AI response display (on main screen, not fully visible here), configurable server/IP settings.

Demonstration: Desktop Application

IntelliGaze - PyQt6 Desktop Client

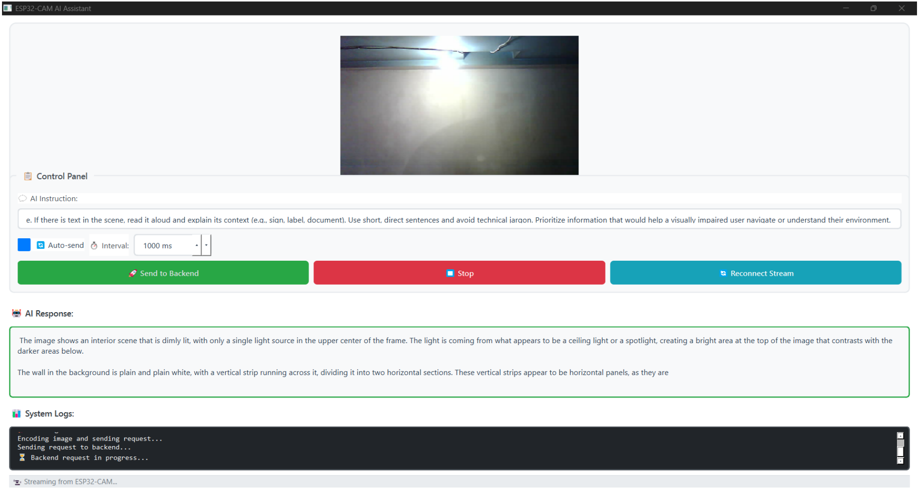
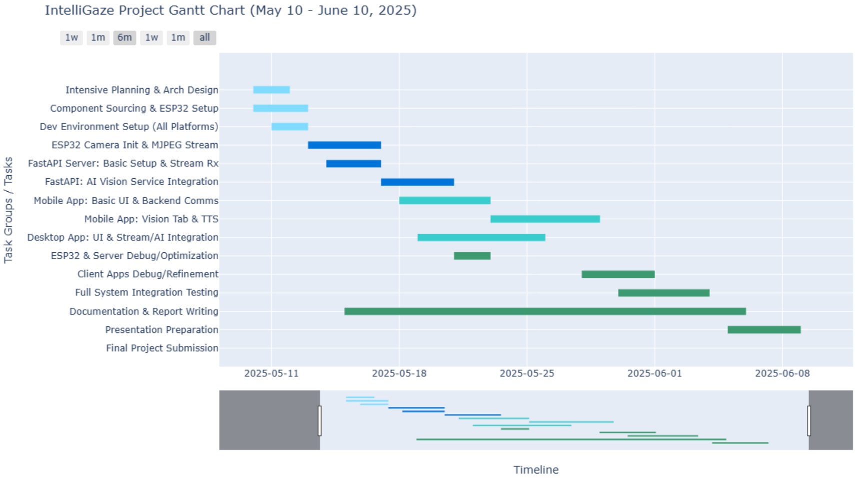


Figure: Desktop interface: Live ESP32-CAM feed, AI instruction input, AI response, and system logs.

Project Timeline: Gantt Chart

IntelliGaze Development Schedule (May 11 - June 10, 2025)

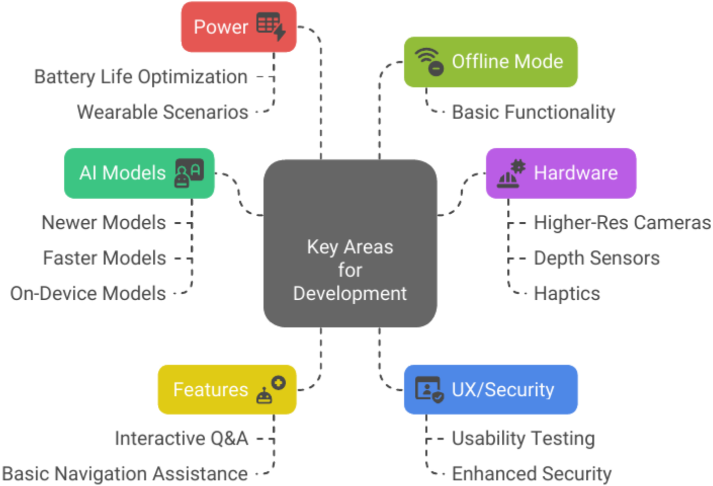


Conclusion

Summary of Achievements

Future Works

Potential Enhancements for IntelliGaze



Team Contributions

Collaborative Efforts in IntelliGaze

- **Touhidul Alam Seyam (230240003):**
  - Project Lead & Chief Architect; Overall system design, development, and end-to-end integration.
  - Lead: React Native mobile app (UI/UX, logic, backend comms).
  - Drove integration of all components; Key troubleshooting; Documentation lead.
- **Shafiul Azam Mahin (230240022):**
  - Lead: PyQt6 Desktop Application (GUI, video feed, direct AI service comms).
- **Eftakar Uddin (230240004):**
  - Lead: FastAPI Backend Server framework (API endpoints, server logic).
- **Muntasir Rahman (230240002):**
  - Lead: ESP32-CAM Firmware (camera init, WiFi, MJPEG streaming).
- **Tasmim Akther Mim (230240025):**
  - Lead: Text-to-Speech (TTS) functionality integration in mobile app (Groq API).

Thank You!

Questions?