LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS6P05NI Project

Final Report

40% Individual Coursework

Submission: Final Submission

Academic Semester: Spring Semester 2025

Credit: 30 Credit Year Long Module

Student Name: Bibek Poudel

London Met ID: 22067316

College ID: NP01AI4A220032

Assignment Due Date: Tuesday, May 6, 2025

Assignment Submission Date: Tuesday, May 6, 2025

Submitted To: Alish KC & Samrat Thapas

I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgement

I extend my sincere gratitude to my internal supervisor, Alish KC, for their invaluable guidance, constructive feedback, and unwavering support throughout this project's development. I also thank my external supervisor, Samrat Thapa, for their expertise and encouragement in refining the system's AI integration. Special thanks to Islington College and the FYP department for providing the infrastructure, resources, and academic environment essential for this research. Their collective contributions were pivotal in achieving the project's objectives, aligning with the collaborative ethos emphasized in acknowledgment best practices.

.

Abstract

This project presents Contextual Object Detection, a system integrating computer vision and natural language processing through the Moondream visual language model. Developed under Islington College's FYP program, the platform enables users to upload images, generate contextual descriptions, and interact via speech-to-text and text-based queries, achieving real-time processing post-GPU optimization (from 20–40 minutes to 4–10 seconds). Rigorous testing validated Django-PyTorch integration, Redis-driven background tasks, and robust authentication workflows, while edge cases like corrupt file uploads highlighted pre-processing vulnerabilities. Cross-device responsiveness and admin dashboard functionalities ensured role-based access and scalable deployment on Linux environments. By merging object detection, scene understanding, and multimodal reasoning, this work advances AI systems capable of contextual inference beyond traditional bounding-box annotations. Findings underscore hardware-software co-optimization's role in reducing latency and enhancing user experience, contributing to evolving practices in deep learning pipelines and interactive AI platforms.

.

Table of Contents

1. Chapter 1: Introduction 1
   1.1. Project Introduction 2
   1.2. Current Scenario in Context in Nepal 3
   1.3. Current Scenario Internationally 4
   1.4. Problem Statement 4
   1.5. Project as a Solution 5
   1.6. Aim and Objectives 5
      1.6.1. Objectives 6
   1.7. Structure of the report 7
2. Chapter 2: Background 9
   2.1. About the end users 10
      2.1.1. Visually Impaired Individuals 10
      2.1.2. Medical Professionals and Researchers 10
   2.2. Requirements Analysis 11
      2.2.1. Python 11
      2.2.2. PyTorch 11
      2.2.3. HuggingFace Transformers 12
      2.2.4. RealtimeSTT 12
      2.2.5. Django 13
      2.2.6. Redis 13
   2.3. Similar Real-World Projects 13
      2.3.1. ChatGPT 13
      2.3.2. Gemini 15
      2.3.3. Deepseek 17
   2.4. Comparison Between Projects 19
   2.5. Similar Research Projects 20
   2.6. Model Architecture: Moondream 21
   2.7. Analysis of Comparison 22
3. Chapter 3: Development 23

3.1. Selected Methodology Explanation 24
3.1.1. Sprint Retrospective: 25
3.2. Phases of methodology 27
3.2.1. Initiation (24th September –4th October 2024) 27
3.2.2. II. Planning ProjectTimeline (2023): October 5–31, 2024 27
3.2.3. System-Architecture Design (Sprint 1: 1 Nov – 15 Nov2024) 28
3.2.4. Sprint 2 (Nov 16–Dec 8): 28
3.2.5. Sprint 3 (Dec 9–Jan 5): 28
3.2.6. Sprint 4 (Jan 6–Feb 1) 28
3.2.7. Sprint 5 (Feb 2–Mar 1) 29
3.2.8. Testing (Sprint 6:Mar 02 – Mar 07, 2025) 29
3.2.9. Deployment (Sprint 7: Mar 8 – Mar12) 29
3.2.10. Reporting (Sprint 8: Mar 13 – April 26, 2025)) 29
3.2.11. Agile Project Tracking Tools 30
3.3. Sprint Backlogs 31
3.3.1. Sprint Review 31
3.3.2. Sprint Summary 31
3.4. Challenges & Adaptations: 32
3.4.1. Adaptations: 32
3.5. Requirement Analysis: 33
3.5.1. User Authentication & Profile Management (AUTH) 33
3.5.2. Image and Scene Processing 33
3.5.3. Visual Query and Interaction 34
3.5.4. Context Generation & Background Task Handling (CTX) 34
3.5.5. Model Management & Extensibility (MMX) 34
3.6. Non-Functional Requirements 34
3.6.1. Performance & Responsiveness: 34
3.6.2. Scalability & Availability: 35
3.6.3. Security & Compliance: 35
3.6.4. Maintainability & Extensibility: 35
3.6.5. Usability & Accessibility: 35

3.7. Design 36
3.7.1. Use case 36
3.7.2. ERD (Entity Relationship Diagram) 37
3.7.3. Data Flow Diagram Level 0 38
3.7.4. Data Flow Diagram Level 1 39
3.7.5. Sequence Diagram Login 40
3.7.6. Sequence Diagram Image Processing 41
3.7.7. Sequence Diagram of System 42
3.7.8. Sequence Diagram of Sequence Query 43
3.7.9. Sequence Diagram of Admin 44
3.7.10. Sequence Diagram of Profile 45
3.7.11. Wireframe Login 46
3.7.12. Wireframe Authentication Page 47
3.7.13. Wireframe Register Page 48
3.7.14. Wireframe of Main Page 49
3.7.15. Wireframe Profile Page. 50
3.8. Implementation 51
3.8.1. System architecture 52
4. Chapter 4: Testing 53
4.1. Agile Sprint-wise Testing 54
4.1.1. Test 1: Test for features from Sprint 3 54
4.1.2. Test 2: To test the speech-to-text feature, user profile and its seamless integration in Sprint 4 58
4.2. Unit Testing, Test Plan 61
4.2.1. Test Case 1: User registration with intended credentials to all fields. 63
4.2.2. Test Case 2: To test login functionality with registered user's credentials. 65
4.2.3. Test Case 3: Verify non-registered user cannot login 67
4.2.4. Test Case 4: Correct login and redirection to main page 69
4.2.5. Test Case 5: Logout and redirection to the authentication page 71
4.2.6. Test Case 6: Check user's access to their profile 73
4.2.7. Test Case 7: Check user can update their profile picture 75

4.2.8. Test Case 8: Confirm whether the user can add bio to their profile. 77

4.2.9. Test Case 9: Check whether the chosen profile picture is updated in profile button of context generation page. 79

4.2.10. Test Case 10: Check browsing the image functionality works. 81

4.2.11. Test Case 11: Confirm that the upload buttons works and context generation starts. 83

4.2.12. Test Case 12: Capture the image in real time and process it for context generation without visual query 85

4.2.13. Test Case 13: Capture the image in real time and process it for context generation with visual query 87

4.2.14. Test Case 14: Consecutive image upload for context generation without visual query 89

4.2.15. Test Case 15: Consecutive image upload and capture for context generation with visual query. 93

4.2.16. Test Case 16: Invalid file upload for context generation 97

4.2.17. Test Case 17: Corrupted Image upload for context generation 99

4.2.18. Test Case 18: Check speech-to-text functionality for user query 101

4.2.19. Test Case 19: Visual query for image without speech-to-text functionality 104

4.2.20. Test Case 20: Short context generation for the image and time taken 106

4.2.21. Test Case 21: Short context and visual query answering for the image and time taken 108

4.2.22. Test Case 22: Terminate context generation. 110

4.2.23. Test Case 23: Create, read/view, update and delete (CRUD) operations for registered users in admin dashboard. 112

4.2.24. Test Case 24: Create, read/view, update and delete (CRUD) operations for generated context and images in admin dashboard.....116

4.3. System Testing, Testing Plan 120

4.3.1. Test 1: Complete User Journey Test 121

4.3.2. Test 2: End-to-End Processing Pipeline Test 126

4.3.3. Test 3: Cross-Feature Interaction Test..... 130

4.3.4. Test 4: Cross Device User Experience Test 132

4.3.5. Test 5: Image processing Pipeline and different image file format 140

4.3.6. Test 6: Running System on Linux Operating System 145

4.3.7. Test 7: To test the context generated from the model on benchmark dataset as ground truth 146
4.3.8. Test 8: To test admin's ability to view admin dashboard and access exclusive features. 151
4.4. Testing Phase Summary 154
4.5. Critical Analysis 155
5. Chapter 5: Conclusion (Legal, Social & Ethical Issues) 156
5.1. Legal Issues 157
5.1.1. Intellectual Property and Copyright Considerations 157
5.1.2. Data Protection and Privacy Regulations 157
5.1.3. AI and Automated Decision-Making Regulations 157
5.1.4. Digital Accessibility Compliance 158
5.2. Social Issues 158
5.2.1. Digital Divide and Technology Access 158
5.2.2. Cultural Context and Representation 158
5.2.3. User Trust and Technology Perception 159
5.3. Ethical Issues 159
5.3.1. Data Privacy and User Confidentiality 159
5.3.2. AI Accuracy and Misidentification Risks 160
5.3.3. Accessibility and Inclusive Design 160
5.4. Advantages 160
5.4.1. Resource-Efficient Visual Intelligence 160
5.4.2. Comprehensive User Experience 161
5.4.3. Multi-Modal Interaction 161
5.4.4. Open-Source Adaptability 161
5.5. Limitations 162
5.5.1. Detection Accuracy Constraints 162
5.5.2. Processing Time and Scalability 162
5.5.3. Technical Deployment Barriers 162
5.6. Future Work 163
5.6.1. Augmented Reality Integration 163

5.6.2. Autonomous Vehicle and Safety Applications 163
5.6.3. Advanced Vision-Language Models 163
5.6.4. Assistive Technology for Visual Impairments 164
6. Chapter 6: References 165
7. Chapter 7: Appendix 167
7.1. Appendix 1: Pre-Survey Form 168
7.2. Appendix 2: Pre-Survey Results 179
7.3. Appendix 3: Post Survey Form 194
7.4. Appendix 4: Post Survey Results 199
7.5. Appendix 5: Important Features and Code Screenshots 206
7.6. Appendix 6: Agile Project Tracking Tool 211
7.7. Appendix 7: Gantt Chart, Work Break Down Structure and Burndown Chart 214

Table of tables

Table 1: Structure of the Report 8
Table 2: Comparison between similar real-world projects 19
Table 3: Agile Test 1: To test the login and context generation feature (sprint 3) 54
Table 4:Agile Test 2: To test the speech-to-text feature, visual query and profile update (sprint 4) 58
Table 5: Unit testing plan 62
Table 6: Test 2: To test login functionality with registered user's credentials 65
Table 7: Test 3: Verify non-registered user cannot login 67
Table 8: Test 4: Correct login and redirection to main page 69
Table 9: Test 5: Logout and redirection to the authentication page 71
Table 10: Test 6: Check user's access to their profile picture 73
Table 11: Test 7: Check user can update their profile picture 75
Table 12: Test 8: Confirm whether the user can add bio to their profile 77
Table 13: Test 9: Check whether the chosen profile picture is update in profile button in context generation page 79
Table 14: Test 10: To check browsing the image functionality works 81
Table 15: Test 11: Confirm that the upload button works and context generation starts 83
Table 16: Test 12: Capture the image in real time and proces it for context geneation without visual query 85
Table 17:Test 13: Capture the image in real time and process it for context with visual query ... 87
Table 18: Test 14: Consecutive image upload for context generation without visual query 89
Table 19: Test 15: Consecutive image upload and capture for context generation with visual qeuery 93
Table 20: Test 16: Invalid file upload for context generation 97
Table 21: Test 17: Corrupted image upload for context generation 99
Table 22: Test 18: Check speech-to-text functionality for user query 101
Table 23: Test 19: Visual query for image without speech-to-text functionality 104
Table 24: Test 20: Short context generation for the image and time taken 106
Table 25: Test 21: Short context and visual query answering for the image and time taken 108
Table 26: Test 22: Terminate context generation 110
Table 27: Test 23: CRUD operations in user details by admin 112
Table 28: CRUD operations on analysed images and context by admin 116
Table 29: System testing plan 120
Table 30: Sys Test 1: To verify the complete user journey in system 121
Table 31: SysTest 2: End-to-End processing pipeline Test 126
Table 32: SysTest 3: Cross-feature integration test 130
Table 33: Cross device configuration test with UI 132
Table 34: SysTest 5: Image processing pipeline and differetn file format 140
Table 35: SysTest 6: Running System on Linux System 145

Table 36: SysTest 7: To test the context generation from the model on benchmark dataset as ground truth 146
Table 37:SysTest 8: To test admin's ability to view admin dashboard and access exclusive features 151

Table of Figures

Figure 1: OpenAI ChatGPT 14
Figure 2: ChatGPT platform 15
Figure 3: Gemini by Google Deepmind 16
Figure 4; Gemini Interface 16
Figure 5: DeepSeek by deepseek 17
Figure 6: Deepseek chat interface 18
Figure 7: Moondream2 Architecture 21
Figure 8:Sscrum methodology 26
Figure 9: Project Tracking and Phases Development in Trello 30
Figure 10: Project Tracking and Phases Planning Trello 30
Figure 11:fig of use case diagram 36
Figure 12:Figure of ERD 37
Figure 13:Figure of DFD 38
Figure 14:Level 1 DFD 39
Figure 15:fig of sequence diagram Login 40
Figure 16:fig of sequence image processing 41
Figure 17:fig of sequence diagram of system 42
Figure 18:fig of sequence of query 43
Figure 19:fig of sequence diagram of admin 44
Figure 20:fig of sequence diagram of profile 45
Figure 21:fig of wireframe of login page 46
Figure 22:fig of wireframe of login and register page 47
Figure 23:fig of wireframe of register 48
Figure 24:fig of wireframe of main page 49
Figure 25:fig of wireframe of profile page 50
Figure 26: System Architecture for the Project 52
Figure 27: Login feature from sprint 3 55
Figure 28: Working login feature 56
Figure 29: Greeted to the system for context generation 57
Figure 30: Successful context generation by the model from sprint 3 57
Figure 31: Enabling microphone for speecht to text 58
Figure 32: Uploaded image 59
Figure 33: Context Generation with visual query and stt 59
Figure 34: Updating the profile picture and bio 60
Figure 35:Registration of new user 63
Figure 36: User registered 64
Figure 37: Filling correct credentials for login 65
Figure 38: Main page after login 66
Figure 39: Login with non-registered user 67

Figure 40: Login failed with unregistered user 68
Figure 41: Login with registered account credentials 69
Figure 42: Successful redirection 70
Figure 43: Logout from logged in user 71
Figure 44: Redirection to authentication page after logout 72
Figure 45: Main page after use log in 73
Figure 46: User access to their profile 74
Figure 47: Choosing the profile picture 75
Figure 48: Profile picture updated 76
Figure 49: Adding bio to the profile 77
Figure 50: Bio updated successfully 78
Figure 51: Main page before updating the profile picutre 79
Figure 52: Context generation page after updating the proifle picture 80
Figure 53: Choosing image for context generation 82
Figure 54: Image chosen successfully 82
Figure 55: Uploading image 83
Figure 56: Image uploading 84
Figure 57: System asking for permission for accessing user's camera 85
Figure 58: Capturing image in real time 86
Figure 59: Generated context for captured image 86
Figure 60: Capturing image in real time with subject 87
Figure 61: Providing captured image with visual query 88
Figure 62: Result for real time capture image with visual query 88
Figure 63: Uploading first image 89
Figure 64: Result for the first image 90
Figure 65: Uploading second image 90
Figure 66: Context for second image 91
Figure 67: Uploading third consecutive image 91
Figure 68: Context for third image 92
Figure 69: Capturing image for visual query 93
Figure 70: Processing first image 94
Figure 71: Context generated for first image 94
Figure 72: Uploading second image 95
Figure 73: Processing second image 95
Figure 74: Context generated for second image 96
Figure 75: Invalid file upload 97
Figure 76: System doesn't allow choosing the invalid file 98
Figure 77: Choosing corrupted image 99
Figure 78: Uploading invalid image 100
Figure 79:System response to corrupted image 100

Figure 80: System response without error to the corrupted image 100
Figure 81: Using microphone for providing visual query through stt 101
Figure 82: User's speech converted to text correctly 102
Figure 83: Processing the image with stt visual query 102
Figure 84: Context generated with answer to user's visual query 103
Figure 85: Uploading image with written visual query 104
Figure 86: Processing the image 105
Figure 87: Generated caption with visual query 105
Figure 88:Uploading the image for short context generation 106
Figure 89: Generated short context and time taken 107
Figure 90: Uploading image with visual query 108
Figure 91: Context and query's answer within 8.24 seconds 109
Figure 92: Uploading image for context 110
Figure 93: Processing the image 111
Figure 94: Terminating the generation process 111
Figure 95: Admin dashboard 112
Figure 96: User's data for CRUD operations 112
Figure 97: Viewing and reading user's information 113
Figure 98: Updating the user's information 113
Figure 99: Updating user's name with new name 114
Figure 100: User's data updated successfully 114
Figure 101: Before deleting the user 114
Figure 102: Deleting the user 115
Figure 103: Deleted the user successfully 115
Figure 104: Recently analysed images in admin dashboard 116
Figure 105: Viewing/reading the analysed image’s information 116
Figure 106:Updating the analysis and context 117
Figure 107: Updating the context with different text 117
Figure 108: Context updated successfully 118
Figure 109: Before deleting the context and image 118
Figure 110: Deleting the context and the image 119
Figure 111: Successfully deleted the context of the image 119
Figure 112: Log in to the system with credentials registered 122
Figure 113: Greeted with context gneration page 123
Figure 114: Uploading image 123
Figure 115: Context generation 124
Figure 116: Visiting profile 124
Figure 117: Updating profile picture 125
Figure 118: Logout from system 125
Figure 119: Uploading landscape image 126

Figure 120: Context for landscape image 127
Figure 121: Uploading lowlight image 127
Figure 122: Generated context for low light image 128
Figure 123: Uploading object heavy image 129
Figure 124: Context for object heavy image 129
Figure 125: Uploading image for context 130
Figure 126: Speech to text for visual query to the image 131
Figure 127: Context and visual query generated 131
Figure 128: Main page in desktop configuration 132
Figure 129: Profile in desktop configuration 133
Figure 130: Profile view in Mobile configuration 134
Figure 131: Main page in tablet configuration 135
Figure 132: Context generation in mobile configuration 136
Figure 133: Caption generated in tablet configuration 137
Figure 134: Uploaing image in tablet configuration 138
Figure 135: Generated context in tablet configuration 138
Figure 136:Users profile in tablet configuration 139
Figure 137: Uploading bmp image 140
Figure 138: Context for bmp image 141
Figure 139: Uploading jpeg image 142
Figure 140: Context for jpeg image 143
Figure 141: Tiff image upload 144
Figure 142: Context for tiff image 144
Figure 143: Running system on linux machine 145
Figure 144: First benchmark image-caption pair 146
Figure 145: Context generation for benchmark 1 147
Figure 146: Context generated for benchmark 1 147
Figure 147: Ground truth image-caption pair for benchmark 2 148
Figure 148: Uploading benchmark 2 image 148
Figure 149: Generated context for benchmark 2 149
Figure 150: Third ground truth and image 149
Figure 151: Generated context for third benchmark image 150
Figure 152:Admin dashboard 151
Figure 153: Admin editing image analysis 152
Figure 154: Admin viewing image analysis 152
Figure 155:Admin deleting the user 153
Figure 156:Admin viewing user details 153
Figure 157:Admin edits user's details 154
Figure 158: Pre survey form-1 168
Figure 159: Pre survey form-2 169

Figure 160: Pre survey form-3 170
Figure 161: Pre survey form-4 171
Figure 162: Pre survey form-6 172
Figure 163: Post survey form-7 173
Figure 164: Post survey form-8 174
Figure 165: Pre survey form-9 175
Figure 166: Pre survey form-10 176
Figure 167: Pre survey form-11 177
Figure 168: Pre survey form-12 178
Figure 169: Pre survey result-1 179
Figure 170: Pre survey results-1 180
Figure 171: Pre survey results-3 181
Figure 172: Pre-survey results-4 182
Figure 173: Pre survey results-6 183
Figure 174: Pre survey results-7 184
Figure 175: Pre survey results-8 185
Figure 176: Pre survey results-9 186
Figure 177: Pre survey results-10 187
Figure 178: Pre survey results-11 188
Figure 179: Pre survey results-12 189
Figure 180 : Pre survey results-13 190
Figure 181: Pre survey results-13 191
Figure 182: Pre survey results-14 192
Figure 183: Pre survey results-15 193
Figure 184: Post survey form-1 194
Figure 185: Post survey form-2 195
Figure 186: Post survey form-4 196
Figure 187: Post survey form-5 197
Figure 188: Post survey form-6 198
Figure 189: Post survey result-1 199
Figure 190:Post survey result-2 200
Figure 191: Post survey result-5 201
Figure 192: Post survey result-6 202
Figure 193: Post survey result-7 203
Figure 194: Post survey result-8 204
Figure 195: Post survey result- 9 205
Figure 196: Views Code-1 206
Figure 197: Views Code-2 206
Figure 198: Views Code-3 207
Figure 199: Settings Code 207

Figure 200: Admin Code 208
Figure 201: Database Model Code 208
Figure 202: Speech to text Code 209
Figure 203: URL Routing Codee 209
Figure 204: Context Generation Model Code 210
Figure 205: Context and Query Code 210
Figure 206: Trello Backlogs-1 211
Figure 207: Trello Backlogs-2 211
Figure 208: Trello Backlog-3 212
Figure 209: Trello Backlog-4 212
Figure 210: Trello Backlog-5 213
Figure 211: Gantt Chart for the project 214
Figure 212: Work Breakdown Structure 215
Figure 213: Project Burndown Chart 215

1. Chapter 1: Introduction

1.1. Project Introduction

“For me context is the key - from that comes the understanding of everything.” – Kenneth Noland

Contextual Object detection is the final year project developed under third year FYP module at Islington College, Kathmandu. It is developed between the intersection of two AI paradigms, computer vision and natural language processing. With the aim of developing a system that can both process and analyze the visual images and scenes and generate the relevant context, this project is powered by a visual language model named Moondream at its core. Increasing complexity of the real world cannot be understood by the traditional computer vision model and downstream tasks such as object detection and instance segmentation. Merely the name of the detected object and localization doesn’t generate sufficient information a system to deal with the real-world edge cases due to which there is a need of a platform and a model which can perform downstream computer vision tasks along with the language modelling ability.

Large language models like GPT, BERT, Mixtral are powerful autoregressive large language model whose abilities are limited up to the text generation. The knowledge and understanding of the language and ability to generate new context from existing text can hugely benefit the artificial intelligence-based system when integrated with visual data and speech data. Contextual Object detection project integrates seamless authentication services for the user, short context generation on uploaded image and captured image in real time, answer to various visual queries, object detection, object pointing, scene understanding, and speech to text query into a single platform. This platform can hugely benefit the users to analyze the images from different sectors such as content creation, medical MRI images, miscellaneous images, and interact with the system using efficient and seamless speech to text engines. Python programming language is the core of this project where the notable frameworks Django and PyTorch powers the web application and the context generation model seamlessly. Redis server is used for handling background context generation task and deal with the cache data. The system also provides the additional feature of profile customization to the users on the system along with the super user privileges to the admin for monitoring and performing CRUD operations on the extracted data. In this way, the project delivers the advanced AI features at the intersections of two paradigms to develop the open-source system for the users to generate, analyze and ask the context related questions to the open-source system. Moreover, the system will allow seamless performance and opens numerous doors for the

development of assistive devices using artificial intelligence and augmented reality along with aspects like autonomous vehicles, factory and family robots and so on. This project is mainly focused on the intersection of computer vision and natural language processing unlike a web application focused on the business-oriented consumers for day-to-day use. Proposed system will go under further development iteratively to develop a framework that integrates the bleeding edge technologies, Artificial Intelligence and Augmented Reality for powering up the embedded systems and robotics.

1.2. Current Scenario in Context in Nepal

Nepal is still in the early stages of using AI tools and technologies and has as such been a laggard in establishing AI regulation (The Kathmandu Post, 2025). The pace of AI development and advancement is still in its nascent stages. Developing artificial intelligence-based solutions and innovations, integrating computer vision and natural language processing requires expensive resources. Nepal have also recognized the transformative potential of AI in education. The government with the vision of 'Digital Nepal for Good Governance, Development, and Prosperity', has endorsed a program of 'Digital Nepal Framework 2019' with a plan of implementation, by five years to make Nepal a digital state in severy sector (Suwal, 2024). Although, the pace of AI-driven solutions is growing in the sectors such as healthcare, education and agriculture, the adoption of resource efficient solutions and foundational models are limited. Majority population in Nepal doesn't have access to state-of-the-art models and platforms due to resource constraints and their expensive nature along with low awareness. Visually impaired individuals are provided with the privileges with basic needs but the potential of aiding their limitations and problems with technology is possible which can address the problem of thousands of people in Nepal (Khatiwada, 2025). Due to the geographical diversity the access to the internet and the resources along with information about the platforms show the potential digital divide between the population in Nepal. In the account of geographical diversity (as a factor of distance from the resources, internet access and so on of the platforms), the information, access and resources and internet connectivity have shown the issue of digital divide among the population of this area.

1.3. Current Scenario Internationally

Internationally, the field of AI has seen rapid advancements since the inception of transformer architecture in 2017. The development process grown exponentially to create extremely capable models like Claude, ChatGPT 4.1, Gemini 2.5 and DeepSeek (IBM, 2025). As reported by UN Trade and Development AI market projected to hit $4.8 trillion by 2033, emerging as dominant frontier technology (UN Trade and Development, 2025). These models can understand the multi-modal data from text, images to sound and videos. Leading tech companies and research institutions have developed multi-modal AI systems that combine language modelling and vision model for spatial understanding in areas like context generation and autonomous driving. IBM Watson Health is another tool developed by IBM for analyzing medical images and generating insights for the doctors (IBM, 2025). DeepMind AlphaFold is another ground-breaking achievement in AI community which combines AI with biology to predict the protein structure leading to drug discoveries and diseases research. In this way, the international stage is actively working on advancement of the AI in every field possible from education to drug discovery which can make our life easier.

1.4. Problem Statement

1. Computer vision downstream tasks such as object detection and instance segmentation provide insufficient spatial information about the environment which makes them very unreliable in real world scenario with complex scenario (Zhong-Qiu Zhao, 2018). Information about the detected objects and the confidence score barely contributes to the complex and challenging real-world problems. These shortcomings really limit how effective computer vision models can be in real-world situations.
2. Most conventional computer vision algorithms lack the ability to combine real-time visual data with text recognition, reasoning, and speech recognition (Sara Abdali, 2024). This gap is particularly evident in tasks that require both understanding and decision-making in complex environments.
3. As infrastructure becomes more complex and technology advances, traditional vision models and language models frequently fall short in providing local and global context (Alexey Dosovitskiy, 2020) about the scene which is hugely essential in areas like autonomous vehicles, medical imaging and facial recognition. These models lack

sophisticated reasoning capabilities (intelligent units) and fail to deliver contextual understanding in real-time, highlighting the need for integrating object detection with more sophisticated reasoning systems.

4. Urgent need of the solution that combines the abilities of computer vision for feature extraction and language model for scene understanding and reasoning along with efficient speech to text conversion for interaction with the intelligent model which can solves the problem in numerous areas like image analysis in medicine, caption generation for content creation, image and scene understanding for visually impaired individuals, autonomous vehicles for language and vision based locomotion system along with AR enabled applications with context generation for real world interactions in most unique way.

1.5. Project as a Solution

1. Contextual object detection project will provide the solution for the limitations of the traditional object detection algorithms by integrating them with open-source large language model as the reasoning unit. This project upgrades the ability of traditional systems into generating the context from the detected frames in real time along with improving the reasoning, analyzing and communicating capability. Concepts like object detection, image captioning, visual question answering, speech to text query conversational AI will provide a detailed understanding of the environment for perception.
2. The fusion of vision encoder and language model will benefit the areas like assistive technology, autonomous navigation, intelligent companion robots and the whole robotics industry. The users will be able to perceive their surroundings more accurately and safely by adding intelligence for real-world survival and adaptability.
3. An open source and efficient 2B parameter visual language model used in the system will efficiently run locally with minimal hardware as compared to the existing tycoons like ChatGPT, Gemini and Claude which require huge number of resources to deploy due to their architectural complexity.

1.6. Aim and Objectives

Contextual Object Detection project mainly aims towards improving the traditional object detection techniques with the integration of large language models for context generation and reasoning for the survival in real-world along with speech to text for querying about the scenarios

in real-world. Django-based web application will bridge the developed pipeline into the intuitive user interface for using this system in actual complex tasks in real-world.

1.6.1. Objectives

1. Unified pipeline development which integrates visual language model, visual query and speech-to-text based visual query to generate context about the image along with the information about the specific query asked about the image.
2. Integrate SigLIP vision encoder and Phi-1.5 transformer-based language model to integrate the 1.86 billion parameter visual language model for context generation.
3. Implement a visual language model optimized for devices with minimal computational power and resources with highly accurate context generation and question answering with efficient speech-to-text functionality.
4. Fuse speech-to-text for seamless interaction with the developed model and system to generate accurate and low-latency contexts.
5. Validate system's reliability in complex scenarios like cluttered environments, low-light conditions through testing and progressive refinement to ensure that the system is ready for real world use cases.
6. Integration of developed and tested AI pipeline with web application for allowing users to access the actual potential of this paradigm for real world use cases.
7. Open sourcing the contextual object detection project for further development and improvement of the project through community support.

1.7. Structure of the report

S.N.	Title	Content	Summary
1	Introduction	• Project Introduction • Current Scenario in Nepal • Current Scenario Internationally • Problem Statement • Project as Solution • Aims and Objectives • Structure of the report	This section includes an opening quote about the context, project overview, the target of the project, an interdisciplinary AI approach, the local and international AI scene, the problem and challenges, the proposed solution and aims and objectives including the technical stacks, implementation, use-cases and future direction of development.
2	Background	• About End Users • Visually Impaired Individuals • Medical Researchers and Enthusiasts • Requirement Analysis • Similar Real-World Projects • Comparison between similar projects • Similar Research Efforts • Model Architecture: Moondream2 • Analysis of Comparison	This section starts out with the definition of target users, i.e., visually impaired medical researchers, and proceeds with requirement analysis, similar real world projects and research efforts, introduces Moondream2 model architecture and its comparative evaluation for the future development.
3	Development	• Selected Methodology • Phases of Methodology • Initiation Phase • Planning Phase • System Architecture Design Phase • Sprint 2 • Sprint 3 • Sprint 4 • Sprint 5 • Sprint 6 (Testing) • Sprint 7 (Deployment) • Sprint 8 (Reporting) • Agile Project Tracking Tool: Trello	This section describes the phases of the agile methodology—from initiation, planning, design till eight sprints that include design and development (Sprints 2–4), testing (Sprint 6), deployment (Sprint 7), and reporting (Sprint 8). It illustrates Trello backlogs, retrospectives, hurdles, adaptations and introduces the system accompanied with use-case diagrams, ERD, sequence diagrams, DFD and wireframes.

		• Sprint Backlogs and Retrospectives • Challenges and Adaptation • Design • Wireframes • Implementation • System Architecture	
4	Testing and Analysis	• Agile Sprint-Wise Testing • Unit Testing, Test Plan • System Testing, Test Plan	This section includes enforcing an agile sprint-wise testing practice, detailing unit testing with its associated test plans, elaborating on the whole system testing, giving a system testing strategy and test plan, controlling iterative quality assurance from component to end-to-end system verification, process coverage.
5	Conclusion	• Legal Issues • Social Issues • Ethical Issues • Advantages • Limitations • Future Work	This section discusses legal, social and ethical aspects of the system, its benefits, limitations, and its next steps including regulatory compliance, societal impact, ethical dilemmas, stakeholder involvement, governance structure, system strengths, weaknesses, together with potential improvements of the proposed roadmap.
6	Appendix	• Pre-Survey Form • Pre-Survey Results • Post Survey Form • Post Survey Results • Important Code Screenshots • Agile Project Management • WBS and Gantt Chart	

Table 1: Structure of the Report

2. Chapter 2: Background

2.1. About the end users

The possible end users for the proposed system are:

2.1.1. Visually Impaired Individuals

Visually Impaired individuals have real issue while gaining the information about the surrounding around them along with the curiosity of knowing about the images and scenes without bothering anyone about explaining it to them. Artificial intelligence addresses this problem through its computer vision paradigms with provided downstream tasks like object detection, instance segmentation. These vision tasks usually gather spatial information about the environment through localization and classification but often failed to address the issue of enough context for a real-world scenario. Contextual Object Detection allows the seamless integration of computer vision with natural language processing to provide the context about the visual images and the surroundings through visually impaired individuals can gather the relevant information to satisfy their hunger of curiosity along with the essence for understanding the real-world environment to survive in day-to-day life. Additionally, the platform provides the functionality to ask questions related to images through which the individuals can ask the questions that comes to their mind without any limitations of artificial intelligence.

2.1.2. Medical Professionals and Researchers

Medical professionals frequently deal with various images from X-rays to MRIs along with microscopic images. Analyzing such images and extracting minute details from them is a tedious task for the long term. Additionally, professionals can focus on more important tasks that require intense attention instead of analyzing numerous images in a day. The platforms like ChatGPT, Gemini provide the ability for their model to analyze and caption the model but they aren't specialized for medical images or for analyzing the images, instead they are for general purpose use case. Contextual Object Detection provides the platform for such professionals who can easily analyze and ask queries about the images and generate the analysis in few seconds reliably. Tumor detection, X-ray analysis, wounds analysis and recognizing medicines can be easily done through the developed project due to which the medical professionals can be hugely benefited with this project.

2.2. Requirements Analysis

Technical Requirements

1. Database: SQL Lite
2. Programming Languages: Python, JavaScript
3. Python Frameworks and Libraries: PyTorch, HuggingFace, Django, Redis, RealtimeSTT
4. Code Editor: Visual Studio Code
5. Interface Design/ Wireframes: Balsamiq
6. Diagram Clarification: Draw.io
7. Version Control: Git and GitHub
8. Project Management Tool: Trello
9. Stable Internet Connection

2.2.1. Python

Python is a high-level, dynamically typed and interpreted programming language which was developed by Guido Van Rossum in 1996. It supports simple syntax and rich paradigms for programming including object-oriented, functional and procedural programming [citation]. Wide use of Python is due to its extensive libraries and frameworks for the development of simple scripts to artificial intelligence. Web development, machine learning, deep learning, automation, data science, robotics are some key areas where Python thrives with its flexibility and rich support. Frameworks including PyTorch for deep learning tasks and Django for web application development are used in contextual object detection with pythonic code. PyTorch provides the rich support to the tensor operations in deep learning along with tasks like dataset preparation, processing, model training, fine-tuning and inference. CUDA cores from the graphics card are utilized through PyTorch for faster computation for loading and inferencing the context generating model. Django handles the user's operations and requests coming from the frontend and efficiently provides the best user experience for a context generation and visual question answering task.

2.2.2. PyTorch

PyTorch is an imperative style, high-performance deep learning library developed by Meta AI Research (Adam Paszke, 2019). It provides the balance between the usability and speech as compared to the deep learning frameworks like Theano (Theano, 2018), Chainer (Chainer, 2025),

Torch. Additionally, PyTorch is very pythonic and supports code as a model, makes debugging easy and is consistent with other popular scientific computing libraries like NumPy and SciPy, while remaining efficient and supporting the hardware accelerators like CUDA from GPU (Nvidia, 2025). It performs immediate execution of dynamic tensor computations with automatic differentiation and GPU acceleration without sacrificing the performance. Extensions like torchaudio, torchtext, torchvision and libraries like hugging face transformers are PyTorch in their core. Downstream tasks simpler as image classification to tasks like visual language modelling, fine-tuning, reinforcement learning are easily implemented using PyTorch due to which it stands as a go-to for prototyping and large-scale deployments of developed AI infrastructure (Adam Paszke, 2019). PyTorch is relatively simple to learn for programmers who are familiar with Python. It offers easy debugging, simple APIs, and compatibility with a wide range of extensions built-in Python. Its dynamic execution model is also excellent for prototyping, although it extracts some performance overhead (Nvidia, 2025).

2.2.3. HuggingFace Transformers

HuggingFace Transformers is a library of pretrained computer vision, natural language processing, audio model for training and inference, datasets, classes required for building AI models (Hugging Face, 2025). Transformers can be used to train the model on custom data, building applications using the models, locally running the generative models, finetuning the models, and using the open-source datasets. Transformer provides simple and optimized inference class for machine learning tasks, trainer class for training and distributed training and generate class for fast text generation through LLMs and VLMs (Keita, 2025). Transformers are specially built for machine learning researchers and educators who wants a framework for using, studying the transformer models along with hands-on practionner to finetune and inference through the large varieties of models (Hugging Face, 2025).

2.2.4. RealtimeSTT

RealtimeSTT is an open-source, robust, efficient and low-latency speech-to-text library with advanced voice activity detection along with instant transcription. This library is available via pip and based on WebRTCVAD AND SileroVAD for voice activity detection, Faster Whisper for GPU accelerated transcription and Porcupine for wake word detection (Beigel, 2025). This library

enables efficient speech-to-text functionality which can be integrated into any application requiring low-latency and very accurate text transcription and wake word detection (Beigel, 2025).

2.2.5. Django

Django is a server-side web framework based on Python. It is a high-level, opinionated Python web framework that provides the rapid development of secure and maintainable web applications (Mozilla Foundation, 2025). With its free and open-source nature along with active community support and helpful documentation, developing the websites becomes easy and straight forward. The hassle of database, admin dashboard, security against vulnerabilities like SQL injection and cross-site forgery are addressed by Django due to which a developer can start development in clear way. Django is platform-independent (Mozilla Foundation, 2025) due to which the applications developed on one operating system can easily run on another with minimum changes (Mozilla Foundation, 2025).

2.2.6. Redis

Redis is a data structures server which provides access to mutable data structures via set of commands, which are based on server-client model for sending commands with TCP sockets and simple protocol (Redis, 2023). Redis is also known as Remote Dictionary Server which implements in-memory key-value database to provide low-latency reads and writes on the data making it suitable for application requiring cache. Redis can be used for building vector database, coaching, inventory management, session management, deduplication and fast data ingest through its solutions and features in enterprise scale (Redis , 2025).

Similar Projects

2.3. Similar Real-World Projects

2.3.1. ChatGPT

ChatGPT is a pioneer large language model to implement the autoregressive nature of language models for daily user's queries and problems related to logic, mathematics, writing stories and code. It was developed by OpenAI in 30 November 2022. Ever since its inception, OpenAI has introduced and provided numerous features and ability to perform well on real world tasks. Advance versions like GPT-4 can analyze and generate the image, enabling the general functionality of visual question answering and image

generation for design purposes. The multimodal capability of ChatGPT makes it very powerful general-purpose tools for day-to-day tasks (OpenAI, 2025).

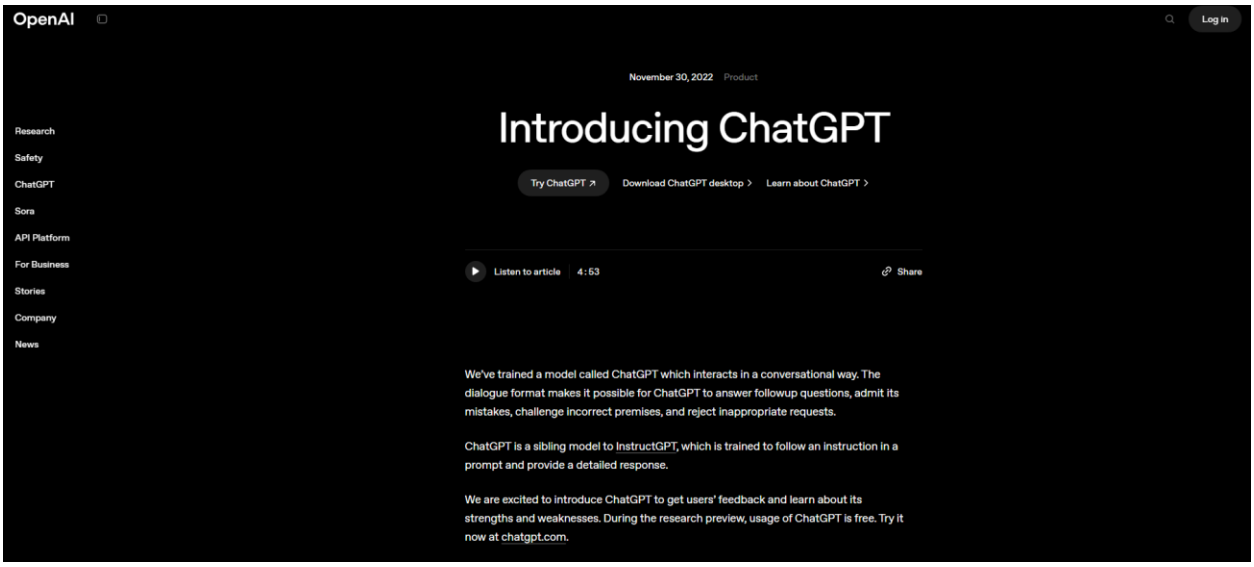


Figure 1: OpenAI ChatGPT

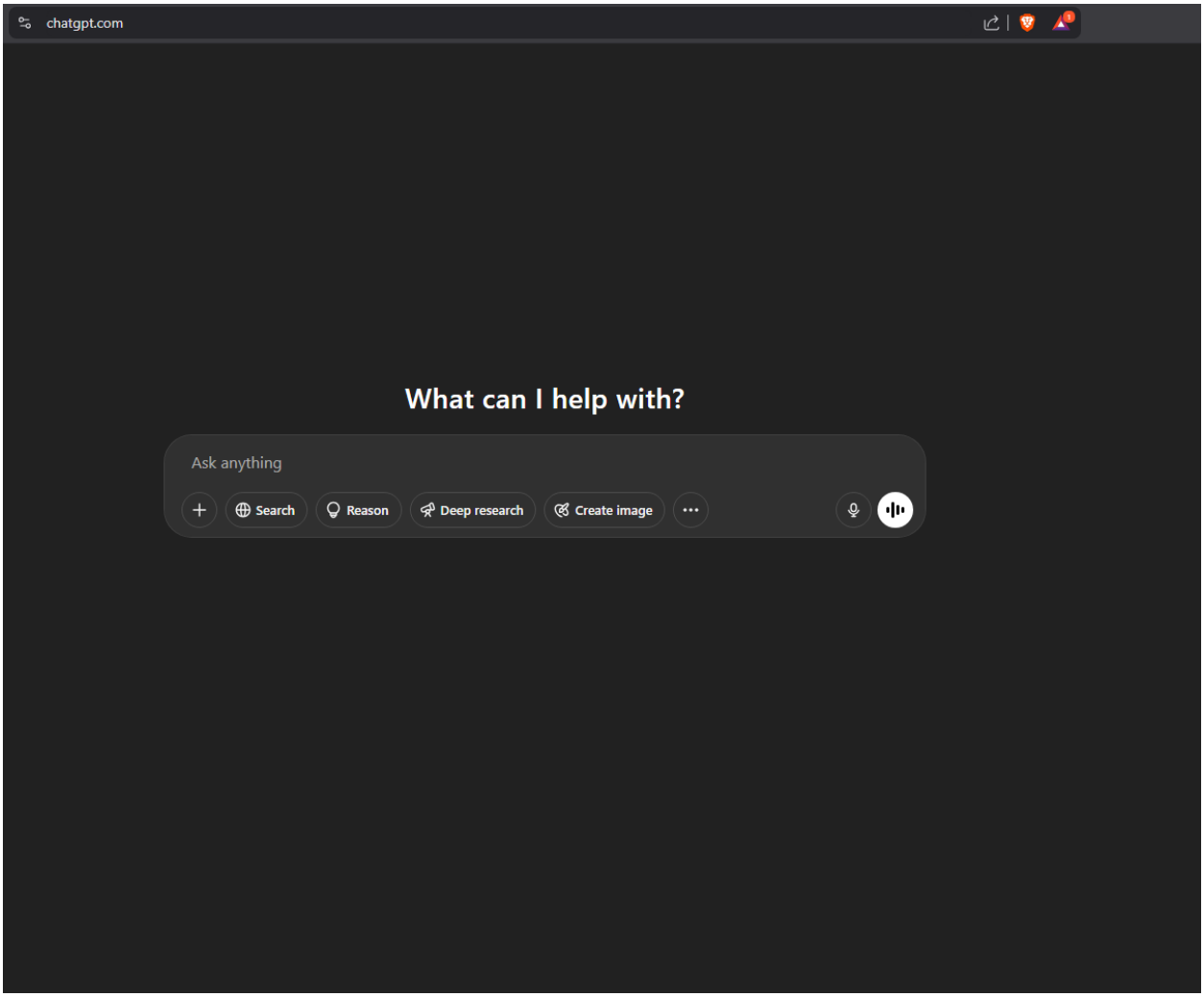


Figure 2: ChatGPT platform

2.3.2. Gemini

Gemini is another prominent large language model developed by Google. Gemini was known as Bard in its early development phases; it was extensively trained by Google on large corpus of data from the web. Gemini have various features such as real time chat and search functionality along with reasoning or thinking capability with their reasoning model for more complex and demanding tasks. Gemini is also a multimodal model which can generate and analyzing text, images, audio and videos. It is commonly used for answering day to day questions, searching through web and refining the search results, generating

code and images and logical and mathematical tasks and ideal for general purpose usages (Google Deepmind, 2025).

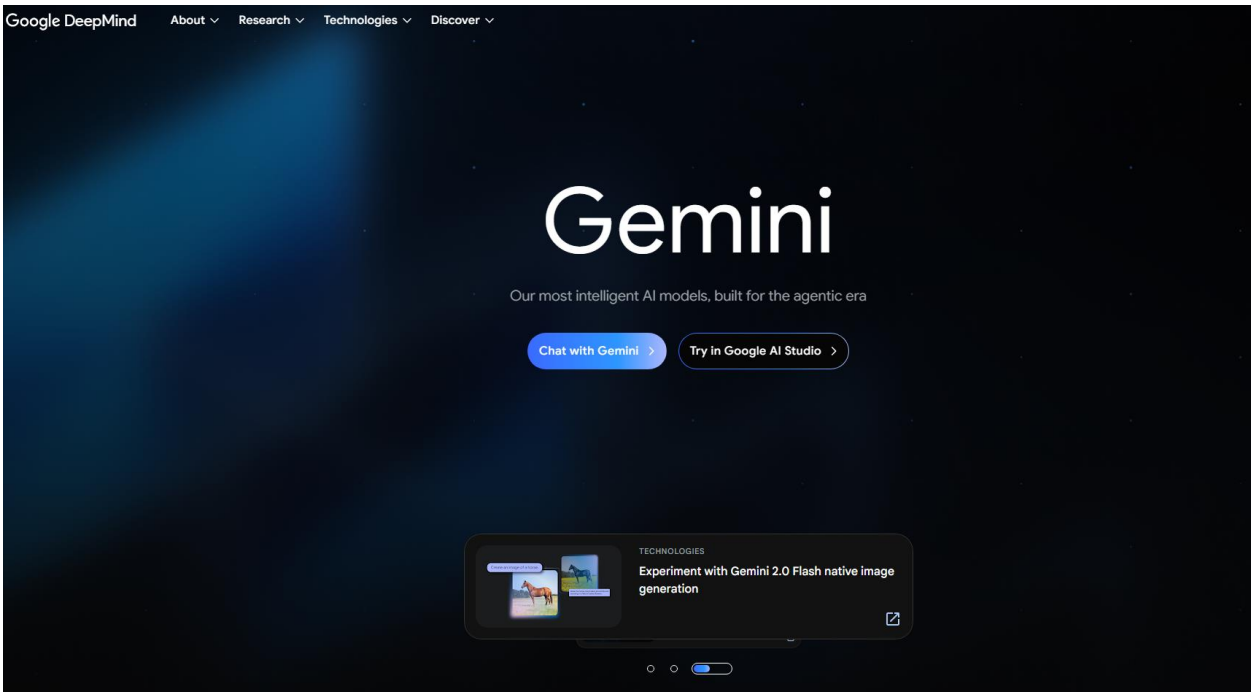


Figure 3: Gemini by Google Deepmind

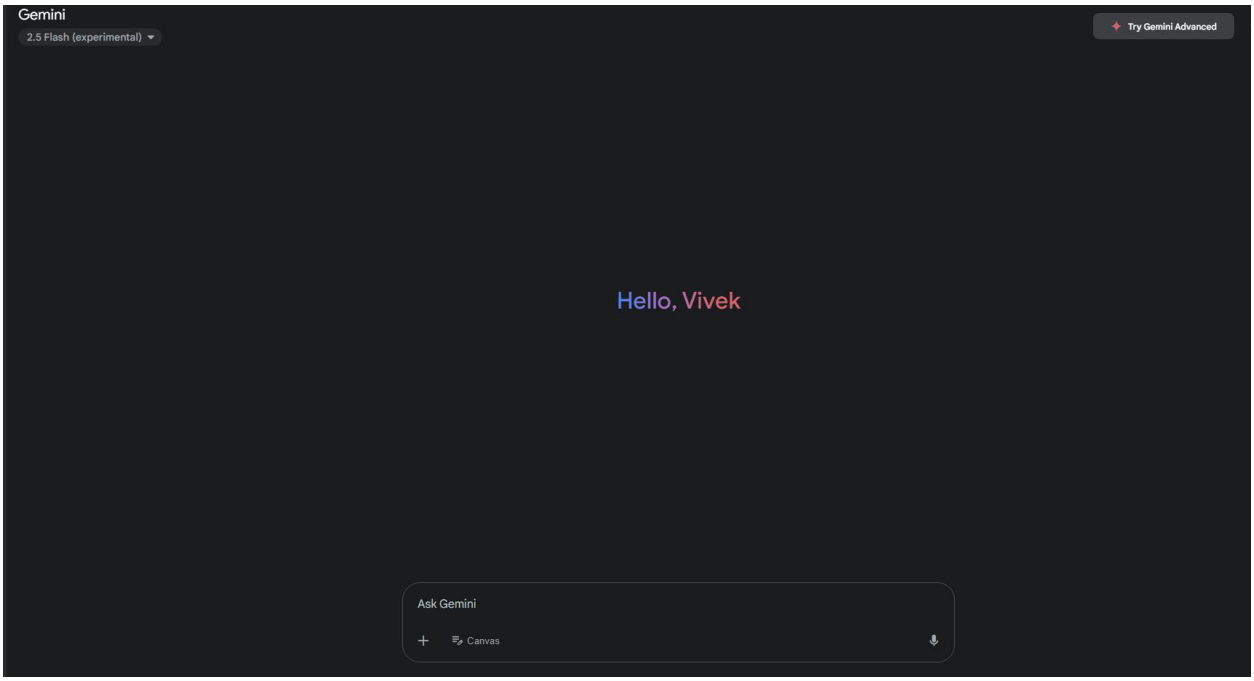


Figure 4; Gemini Interface

2.3.3. Deepseek

Deepseek is an open-weight model developed by DeepSeek in 2024. It is a platform with series of large language models focused on text generation, reasoning and code generation. The parameter counts ranges from 1.3B to 67B for the open-source reasoning and visual models. Deepseek also provides specialized model for coding with the priority in quality code generation. Being opensource and a newcomer in the industry of large language models and chatbots, models from DeepSeek provided comparable performance to its contemporary competitors like Google's Gemini, OpenAI ChatGPT and Anthropic Claude. DeepSeek incorporated advance machine learning techniques such as reinforcement learning, mixture of experts (MOE) which enables it to perform in same level as its competitors in tasks such including mathematical reasoning, logical reasoning, code generation, daily queries and images analysis (DeepSeek, 2025).

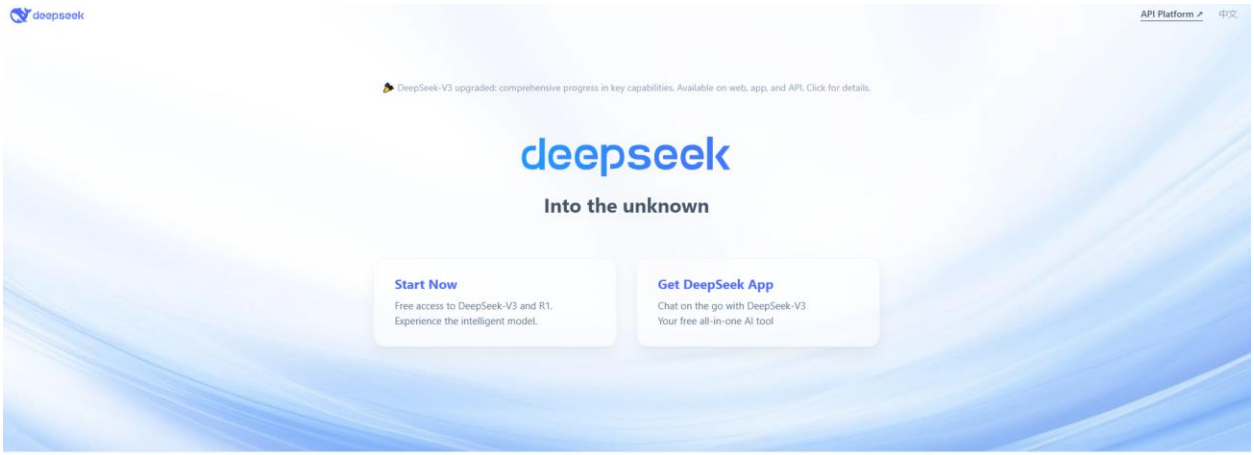


Figure 5: DeepSeek by deepseek

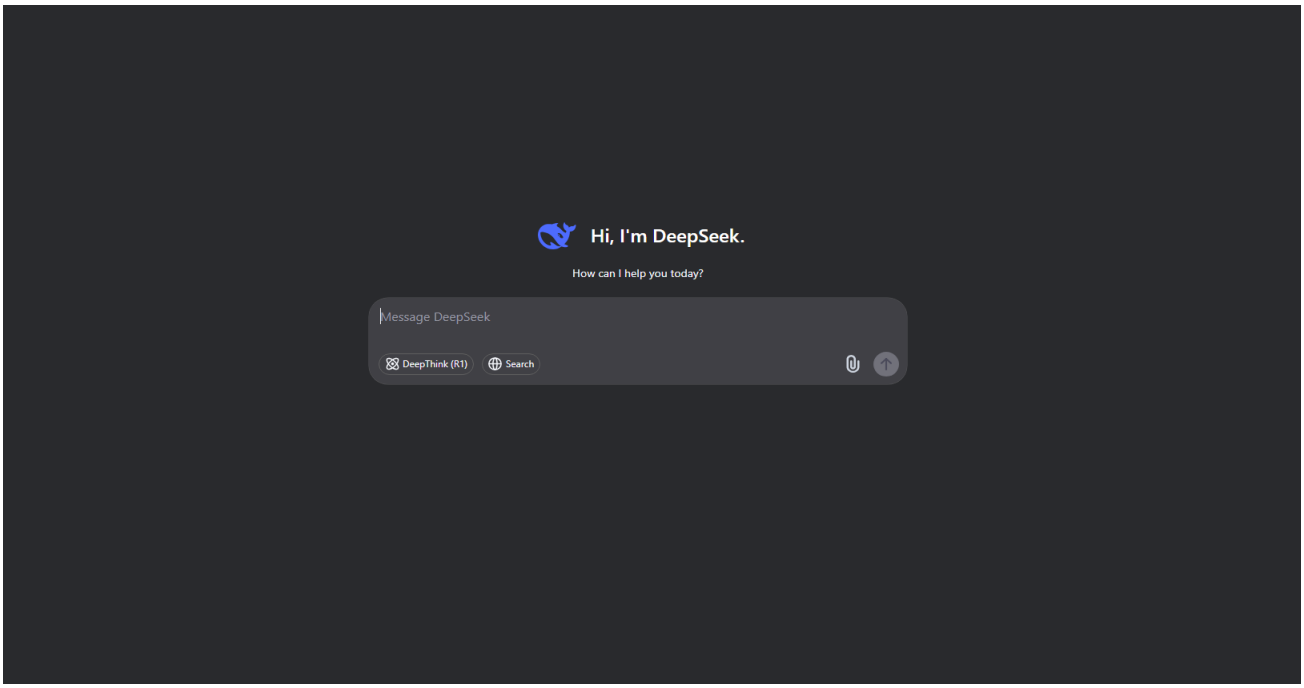


Figure 6: Deepseek chat interface

2.4. Comparison Between Projects

The table below shows the comparison between the proposed system and a similar project based on their features.

S.N.	Feature	ChatGPT	Gemini	DeepSeek	Contextual Object Detection
1	Text Generation	✓	✓	✓	✓
2	Image Understanding	Limited	✓	Limited	✓
3	Object Detection	×	Limited	×	✓
4	Image Captioning	Basic	Good	Basic	✓
5	Visual Q&A	Limited	✓	Limited	✓
6	User Authentication	✓	✓	✓	✓
7	Real-time Camera	×	×	×	✓
8.	Speech-to-Text	✓	✓	×	✓
9.	Detection History	×	×	×	✓
10	Admin Dashboard	Limited	Limited	Limited	✓
11	Background Processing	Hidden	Hidden	Hidden	✓ (Redis)
12	Custom Model Support	×	×	×	✓
13	Open Source	×	×	×	✓
14	Dedicated Visual Context Generation	×	×	×	✓
15	Computational Resources	Extreme	Extreme	Medium	Low

Table 2: Comparison between similar real-world projects

2.5. Similar Research Projects

1. ContextDet: A unified multi-modal model called ContextDet has been developed which is capable of end-to-end differentiable modelling of visual-language contexts, to locate, identify, and associate visual objects with language inputs for human-AI interaction. This work proposed the concept of generate-then-detect which enables users to detect object words within human vocabulary. ContextDet mainly includes three main components: (a) visual encoder that extracts both global and local visual representations, (b) multi-modal large language model for generating the context that integrates language with visual inputs, (c) a visual decoder to predict the object locations and bounding boxes. This combination enabled the system to perform accurately in diverse settings. By utilizing multi-modal language model potential with the generate-then-detect approach, this work has provided the possibilities in the aspects from autonomous systems to interactive robots leading towards the object detection for human-AI interaction and empowering models to handle dynamic and real-world scenarios with great understanding (Yuhang Zang, 2024).
2. Grounding DINO: Grounding DINO, an open-set object detector in the paper Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection, combines the Transformer based DINO detector designed and developed by Facebook AI Research team through the paper Emerging Properties in Self-Supervised Vision Transformers (Facebook AI Research, Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, Armand Joulin, 29 Apr 2021). It allows detection of novel objects based on human inputs, such as category names or referring expressions. This paper presents three key steps of object detection algorithm: (a) closed-set detection with predefined categories, (b) evaluation of novel objects and referring expression comprehension benchmarks, and (c) an image editing application that merges Ground DINO with Stable Diffusion (Shilong Liu, 2023). This approach involves a fusion of language and vision modalities through features like feature enhancer, language-guided query selection and cross-modality decoder for achieving strong results on multiple benchmarks.

2.6. Model Architecture: Moondream

Moondream2 is the lightweight VLM with 1.86B parameters that are obtained from the SigLIP (image encoder) and Phi-1: it is an open-source vision language model (VLM) for constrained applications in microcontrollers. 5 (language decoder) (Verma, 2024). The SigLIP is inspired by a pairwise and sigmoid loss for enhancing zero-shot classification, while the Φ-1 attitude loss huri2018attitude attends only to the top 1 class. 5 Verma++, a 1.3B param transformer, which specializes in multi-step reasoning and knowledge understanding (Verma, 2024). Transparency The code transformer, Timm and einops are used to be able to do tasks like image captioning, visual question-answering, Optical Character Recognition (OCR), and Commonsense Reasoning. Even though it still has room for improvement in abstract and fine-grained reasoning, Moondream2 shows good generalizability with practical applications, such as retail analytics and robotics, thereby showing the progress of efficient multimodal systems. Its light-weight design strikes a balance between pragmatics and performance by providing scalable solutions for resource-constrained environments and demonstrates the problematics of VLM in maturity (Verma, 2024).

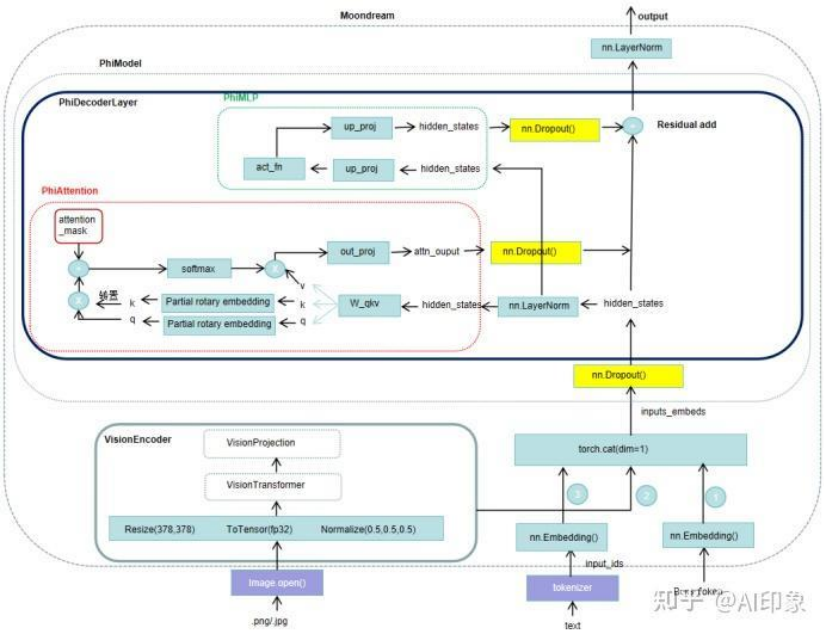


Figure 7: Moondream2 Architecture

2.7. Analysis of Comparison

Contextual Object Detection Framework is a dedicated, low-cost solution for enterprises looking for visual analytics rather than broad-purpose AI. Focusing its architecture to only the domain-specific visual understanding viability, the model inevitably enables greater performance with a slight footprint making it ideal for quick deployment under low-end applications. While fully featured AI assistants may depend on complex external APIs, the system does include an end-to-end authentication system, admin interfaces, and activity logging, which means no additional development work is required. Tailored configurations, on-premises processing capabilities, and multimedia inputs (still images, real-time camera streams, or voice instructions) usability create an integrated system that meets a range of operational needs and accessibility expectations, making it the most adaptable model possible. These qualities are beneficial when dealing in areas with unstable networking frameworks or harsh data governance regulation (a couple of developing regions), where an expansive AI framework is by and large unrealistic.

This bottom-up method has a logical foundation along-due to its narrow-based of varieties. Its small computational footprint allows it to be effectively deployed in low resource environments—low resource hardware (e.g. remote field sites), and low resource urban settings (e.g. Nepal). The platform bypasses the performance limitations of multipurpose AI systems and attains a higher level of accuracy in executing specific tasks by focusing solely on visual analysis. Further its open-source software distribution of the framework and its support for custom model architectures allow enterprises to retain data sovereignty and avoid vendor lock-in on commercial cloud offerings, in line with the ever-growing importance of transparency and governance in recent developments. With the addition of speech-to-text and visual processing capabilities, users with different modalities of interaction can be included as well. Together, these traits demonstrate how they offer powerful, scalable, low-friction, and low-resource alternatives among resource-light stacks appropriate for innovation- and tech-starved or emerging-market settings.

3. Chapter 3: Development

3.1. Selected Methodology Explanation

The project selected Scrum as its project management framework because Scrum is well-suited for the iterative development nature of research and model training and system integration. Scrum allows for collaboration during work periods that are organized by timeboxed intervals, aiding in the delivery and review of distinct functional backends (e.g., data collection pipelines and model-serving APIs) at user-defined intervals, while still being able to incorporate new insights and stakeholder input immediately**.**

A scrum two-week sprint means we must do all the key ceremonies.

1. **Sprint Planning:** Start by ordering the items in our backlog by how closely they correspond to the stages specified in our Gantt chart: “Define project scope,” “Gather requirements,” “Prototype model training,” “Integrate OCR and voice modules”. They are all in a user story format: "As a researcher, I want to preprocess annotated images, enabling model trains free of data errors", with acceptance criteria well defined and a rough estimate of the story points.
2. **Daily Stand-ups:** Each day the team rallies to review progress (e.g., “Dataset cleaned to 90% completeness”) and blockers (e.g., “GPU access delayed”) with specific next steps. A fast check-in process allows the team to identify and address all manners of blockers, from hardware to data quality to algorithmic issues.
3. **Sprint Review:** At the end of each sprint a working increment is shown which should have included the milestones of the Gantt chart:
4. **Sprint 1 (System Architecture Design):** UML diagrams, data-pipeline prototypes, and stubbed REST endpoints.
5. **Sprints 2-5 (Development):** An increasingly smart object-detection framework: train YOLO or Faster R-CNN on meticulously chosen datasets, incorporate vision-and-language captioning, OCR, and voice commands in our React-based UI..
6. **Sprint 6 (Testing):** As per the results from the Unit, Integration and performance tests, inference time is less than 200 milliseconds for concurrent loads.
7. **Sprint 7 (Deployment) & Sprint 8 (Reporting):** Live deployment to Vercel/Render with GPU backed model servers, as well as draft documentation screenshots.

3.1.1. Sprint Retrospective:

Every sprint is closed out with an accounting of what went right (for example: “Our preprocessing pipeline cut annotation errors by 30%”) and what didn’t (for example: “We need to secure GPU time ahead of time to avoid model-training delays”). Tangible changes — like speeding up how to deploy models in our CI/CD or adding additional annotation-quality checks — are included in the next sprint (Sprint n + 1) planning.

Through the introduction of risk management, continuous stakeholder engagement, and regular delivery in the 2-week cadence, Scrum supports us in untangling the complex dependencies of contextual object detection — where model precision, data quality and real-time UI integration all need to move forward together. This feedback-based agile process ensures to us that every stage of our Gantt chart (from “Project Initialization” to “Reporting”) delivers concrete, tested results, that eventually gives as a strong, production-level object detection system.

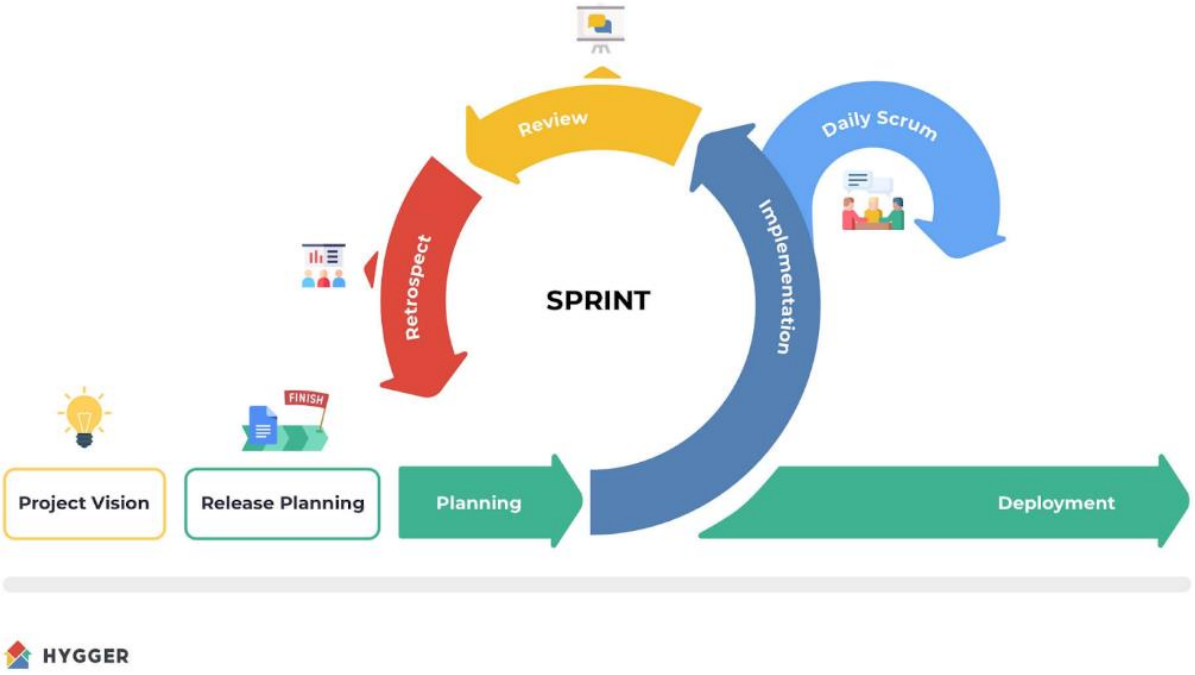


Figure 8:Sscrum methodology

3.2. Phases of methodology

3.2.1. Initiation (24th September – 4th October 2024)

The initiative began with a ten-days Initialization phase, whose purpose was to work out the project's vision, scope and technical feasibility. From September 24-26 we took part in the kickoff workshops, to outline the high-level goals and what would the "Contextual Object Detection" have to achieve in terms of accuracy, latency and target use-cases. On the 27th to 29th September requirements gathering sessions with domain experts and end-users, we identified functional (e.g., object classes, annotation formats) and non-functional (e.g., compute budget, points of integration) requirements effects. On September 30 and October 1, we researched potential toolchains including Python vs C++ libraries, supported hardware accelerators, and data set origins, and on October 2 we conducted an exception analysis to predict edge-cases (e.g. low-light images and occlusions). And then, on October 3 to 4, we prepared a feasibility report that summarized what we had found and was approved to go forward.

3.2.2. II. Planning Project Timeline (2023): October 5–31, 2024

Once the feasibility was confirmed, there was a four-week period of substantial planning before submitting a proposal. Between October 5-8 we built a complete project plan in gantt which broke down the roadmap into six individual sprints that would culminate in beta and used ideal hours to show task dependencies and estimated times. Resource allocation on October 9–11 filled the roles of Data Engineer / ML Researcher and Front-end Developer and acquired hardware that has been tested. A working group on October 12–14 evaluated these higher-level risks and caused mitigation strategies to be put in place. User flow testing was performed in October 15–18 on a Figma prototype integrated with simple mockups. We did Milestone 1 (Supervisor Approval) on October 19 and 20-23 focused on finalizing the formal proposal document. We submitted Milestone 1.1 between October 24–31 after a round of revisions we received from our advisors.

3.2.3. System-Architecture Design (Sprint 1: 1 Nov – 15 Nov 2024)

Architecture design and Backlog grooming were the main topics in the first iteration sprint. On Nov 1–5 we did research and consumed representative datasets — COCO subsets, custom annotated frames — and laid out skeleton of the core data pipeline. The team worked on equipment and infrastructure setup and the UML (Unified Modelling Language) diagrams for the system components (data loader, model server and UI client) from Nov 6-9, and created API contracts for model serving. A mid-sprint backlog review (Nov 10–11) that clarified acceptance criteria for all the stories. Finally, we finished Milestone 2 on November 12-15, where we submitted sequence, class, and deployment diagrams for formal sign-off. Development (Sprints 2–5: November 16, 2024 – March 1, 2025) We developed and iterated upon the object-detection pipeline and UI over the course of four concurrent sub-sprints.

3.2.4. Sprint 2 (Nov 16–Dec 8):

Dataset collection/preprocessing, interim report write-up, and initial model-training scripts (YOLO, Faster R-CNN) were implemented. A postmortem winter backlog review guaranteed to get us training on GPU instances by Dec 1.

3.2.5. Sprint 3 (Dec 9–Jan 5):

Front-end components to upload an image and display a bounding box in real time went live. We wired up our first trained model endpoints behind a Fast API server, ran model-evaluation benchmarks, and honed our training curves.

3.2.6. Sprint 4 (Jan 6–Feb 1)

Integrated the vision and language modules – generated object-label descriptions through a transformer based captioning API and wire the web client with these services. Bug Fixes and Voice Command Prototypes were merged (and backlog reviewed to chase Milestone 3).

3.2.7. Sprint 5 (Feb 2–Mar 1)

Incorporated responsive UI tweaks, adjusted inference latency for performance, and completed integration tests. By the end of Sprint 5 we had a working demo with real time video object detection, a speech to text and visual overlays of descriptions.

3.2.8. Testing (Sprint 6: Mar 02 – Mar 07, 2025)

In this week´s long sprint we performed lots of testing. Unit and component tests to validate each frontend widget and backend endpoint from March 2–4 and then performance testing under load (up to 50 concurrent video streams) on March 5–6. A last Sprint 6 backlog review on March 7 marked the achievement of Milestone 4: a codebase that is fully tested and regression-free and ready to be deployed.

3.2.9. Deployment (Sprint 7: Mar 8 – Mar 12)

Over 5 days we dockerized each service and implemented CI/CD automation along with the deployment of the application. The deployment included the use of Vercel servers for React client implementation and Render servers with GPU enabled instances for Fast API and model servers, and we used a managed cloud service for MongoDB database hosting. Sprint 7 backlog review and Milestone 5 “Deploy the Application” sign-off – 12 March – application for deployment has been approved by the QAPP leadership.

3.2.10. Reporting (Sprint 8: Mar 13 – April 26, 2025))

Phase 7: Seven weeks of documentation and project closure. From March 13–April 5, we completed writing the final report (introduction, methodology, architecture, development, testing, ethical considerations) and were putting together appendices (sample code, screenshots, survey results). Monitoring of platform metrics, including any post-deployment tweaks, and compilation of MAA evidence took place from 6–20 April. Between April 21 and May 2, we conducted stakeholder presentations, a project retrospective, wrapped up all deliverables, and officially closed out with Milestone 6 “Finalize the Documentation.”

3.2.11. Agile Project Tracking Tools

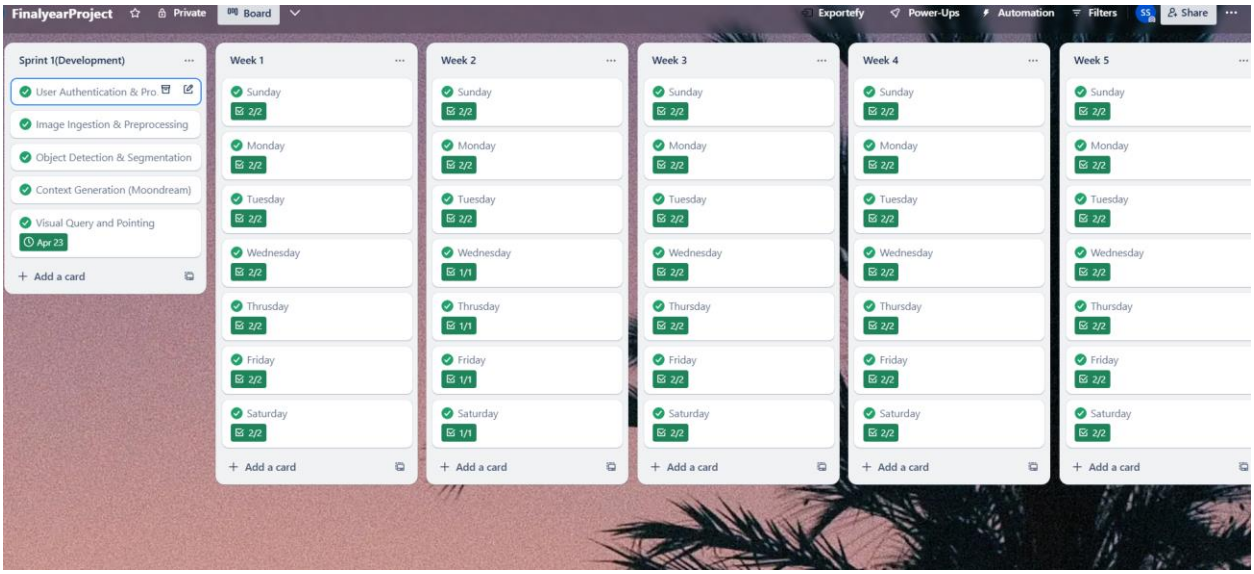


Figure 9: Project Tracking and Phases Development in Trello

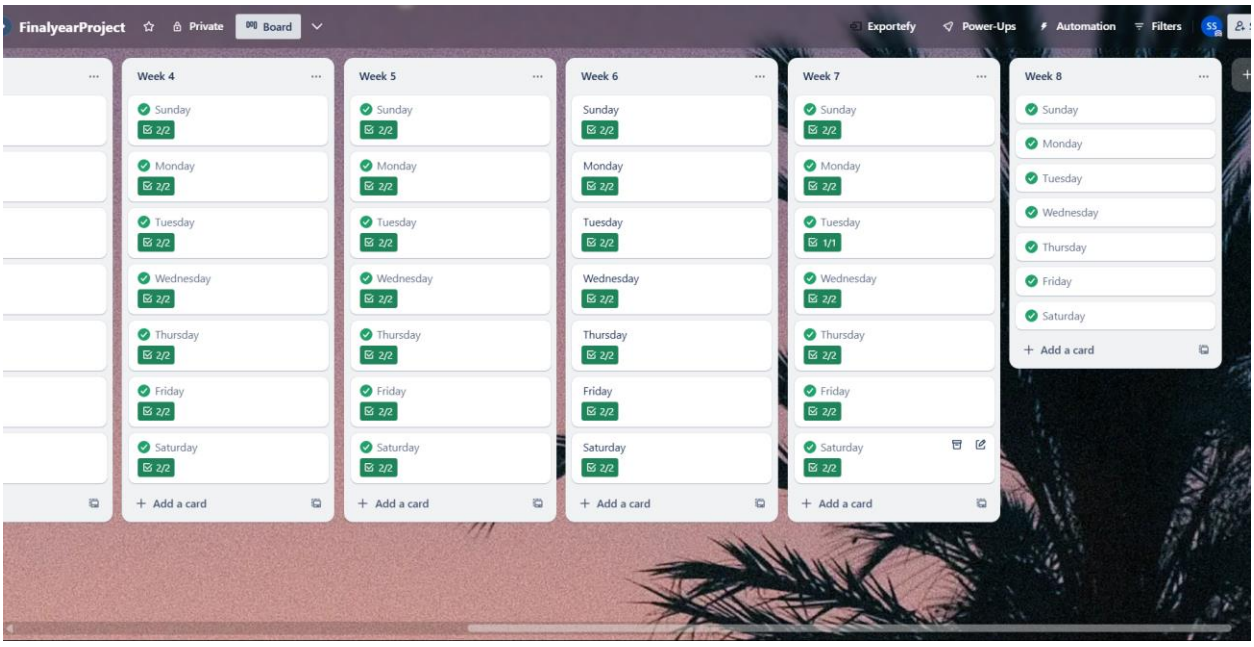


Figure 10: Project Tracking and Phases Planning Trello

3.3. Sprint Backlogs

Regarding the Gantt chart, the Sprint Backlog is a set of tasks that are set for the duration of every single Sprint to meet the specific objectives defined for that time-boxed period. The chart not only displays that for each Sprint, other work is being done, it also shows that each Sprint is unique, hence the contents of the Sprint Backlog will be different Sprint per Sprint. such as 'SPRINT 1' under 'System Architecture Design' consists of working 'Dataset research and Collection', 'Research and development of mod...," and "UI / UX for web application." These activities are the work items from the Sprint Backlog that the team has agreed to complete. The graph says that the Sprint Backlog is made in the planning of each Sprint.

3.3.1. Sprint Review

In a Gantt chart You can see the "Sprint Backlog Review" is a task for many Sprints along the phases such as "System Architecture design", "Develop", "Testing" and "Deploy". This suggests that there is a review activity on the Sprint Backlog in these Sprints. But we don't see a "Sprint Review" on the chart. Sprint Review is a meeting that the development team conducts at the end of the sprint to show the increment and elicit feedback. The chart neither indicates what is covered in the "Sprint Backlog Review", or who is present.

3.3.2. Sprint Summary

The project itself is divided into several sectors: System Architecture Design, Development, Testing, Deployment and Reporting. System Architecture Design: Our work started with dataset research and collection then research and development for models, and UI/UX design to lead to a System Architecture milestone. The Development phase proceeded in an orderly manner, including collection of data sets, pre-processing, training of the Object Detection and Language Model, and initial stages. Then the UI design was implemented with a custom model architecture. Then we added vision and language models to the web application. The development ended up in development and integration of OCR and voice capabilities. After the development, Testing comprised full functional and performance testing, particularly for the vision models. A Deploy phase concentrated on containerizing and deploying the developed system and marked a deployment milestone. Lastly, the Reporting stage focused on document writing, progress monitoring, evidence for comparison selection, project review report writing, and the document closure. Routine backlog reviews were performed in each of these stages.

3.4. Challenges & Adaptations:

- Problems to connecting AI models with the implemented web application.
- Large language models with big local load times.
- Slow progress on GUI works to handle Moondream2 outputs.
- Crashing frequently and lag on local load/query of LLAMA.
- Problems with installation of CUDA Toolkit, which led to delays in the GPU support.

3.4.1. Adaptations:

- Decided to try Moondream2 as the model that might be accurate and computationally efficient model for prototyping.
- Ollama: Opening large language models and making them accessible locally using small memory.
- Learned something new about the Django framework and got the development environment set up.
- Built a quick prototype with transformer model integration and a working frontend built on Django.
- Successfully installed a PyTorch at a safe level in laptop with 6GB V-RAM.

3.5. Requirement Analysis:

Requirement Analysis for Contextual Object Detection Details the detailed capabilities and constraints necessary to implement the project vision of "a seamless platform that incorporates computer vision and language understanding" into a system. It uses stakeholder interviews (faculty mentors, end-users who create content for and use medical imaging), technology evaluations (Django, PyTorch, Redis), and use-case workshops to discover, validate, and capture both functional and non-functional requirements.

3.5.1. User Authentication & Profile Management (AUTH)

Functional Requirements

- REQ-AUTH-1: User Registration Users should be able to sign up for the website and log in using a secure username/password flow using Django's authentication system.
- REQ-AUTH-2: User profiles will be loaded from the database with their preference language, UI theme and usage history on each login.
- REQ-AUTH-3: The super-users admin (administrators) shall be able to C.R.U.D (Create, Read, Update, Delete) create, read, update, and delete user account or the user profile customization using the Django admin panel.

3.5.2. Image and Scene Processing

Functional Requirements

- REQ-ISP-1: The platform shall receive a variety of image sources, provided both as uploaded images and as live camera content, and pass them to the Moondream visual-language model for assessment.
- REQ-ISP-2: The system must detect and segment all objects detectable in the frame within 5s of submission.
- REQ-ISP-3: The model must produce a short natural-language "context summary" of the scene—its objects, their relationships, and activities it has inferred—and return it over a REST endpoint.

3.5.3. Visual Query and Interaction

Functional Requirements

- REQ-VQI-1: The user should be able to click (say) to indicate an object that has been perceived, and the system should report the name of the object and the relation between such an object and the other objects in the scene.
- REQ-VQI-2: The system should allow free-form visual queries using speech-to-text: users query "What is the person holding?" and get an answer in plain English.
- REQ-VQI-3: All interactions (pointing, queries, context-requests) should be logged with timestamps, users of all audit and model-improvement

3.5.4. Context Generation & Background Task Handling (CTX)

Functional Requirements

- REQ-CTX-1: Context-aware summaries and answers should run asynchronouly in Redis-based task queues so that the web UI page remains available to the user.
- REQ-CTX-2: The task-status endpoints SHOULD specify the state of progress (queued, processing, completed) so that UI can perform polling and display real-time updates.
- REQ-CTX-3: The system is to cache most recent context results in Redis for 10min, to speed up subsequent queries on the same image.

3.5.5. Model Management & Extensibility (MMX)

Functional Requirements

- REQ-MMX-1: Django admin interface should allow admins to upload new models checkpoints (Moondream variants or fine-tuned weights) and change the current model in use, without code changes.
- REQ-MMX-2: The system shall validate model files that are uploaded (size, format) and revert to the previous version if the upload failed.

3.6. Non-Functional Requirements

3.6.1. Performance & Responsiveness:

- NFR-P-1: On-the-fly (camera feed) image processing (detection +context) should be under 8 seconds/frame.

- NFR-P-2: Duration of speech-to-text transcription and natural-language response shall be instantaneous (3 s).
- NFR-P-3: Page load performance on dashboard pages (such as profile, logs or admin pages) shall render in under 2 seconds under normal load.

3.6.2. Scalability & Availability:

- NFR-S-1: A minimum of 100 concurrent users simultaneously engaging in mixed tasks (upload, query, admin) with 99% availability.
- NFR-S-2: Redis task queue workers and Django web processes should scale via horizontal scaling based on CPU usage thresholds.

3.6.3. Security & Compliance:

- NFR-C-1) User Credential, images and model artifacts – At rest and in Transit (HTTPS, Django cryptographic setting) All user credentials, images and model artifacts must be encrypted at rest and in transmit.
- NFR-C-2: Access control shall be enforced using role-based permissions—with the exception that model upload or user management actions are to be carried out by super-users only.

3.6.4. Maintainability & Extensibility:

- NFR-M-1: AI modules (for vision, language, authentication) shall be implemented in the form of a plug-in to be able to integrate new AI frameworks in the future.
- NFR-M-2: All public endpoints shall have well-documented RESTful APIs (Swagger/Open API) which shall be consistently maintained and available.

3.6.5. Usability & Accessibility:

- NFR-U-1: The web UI must meet the WCAG2.1 AA; all function shall be accessible using only the keyboard.
- NFR-U-2: There should be visual feedback (loading spinners, progress bars) to direct users during long-running tasks (context generation, model switching).

This need finding validation provides the assurance that the Contextual Object Detection platform is constructed on a strong stakeholder-validated foundation - as an intersection of live computer

vision, natural-language comprehension, and robust management - to provide a compelling, manageable, and secure AI tool.

3.7. Design

3.7.1. Use case

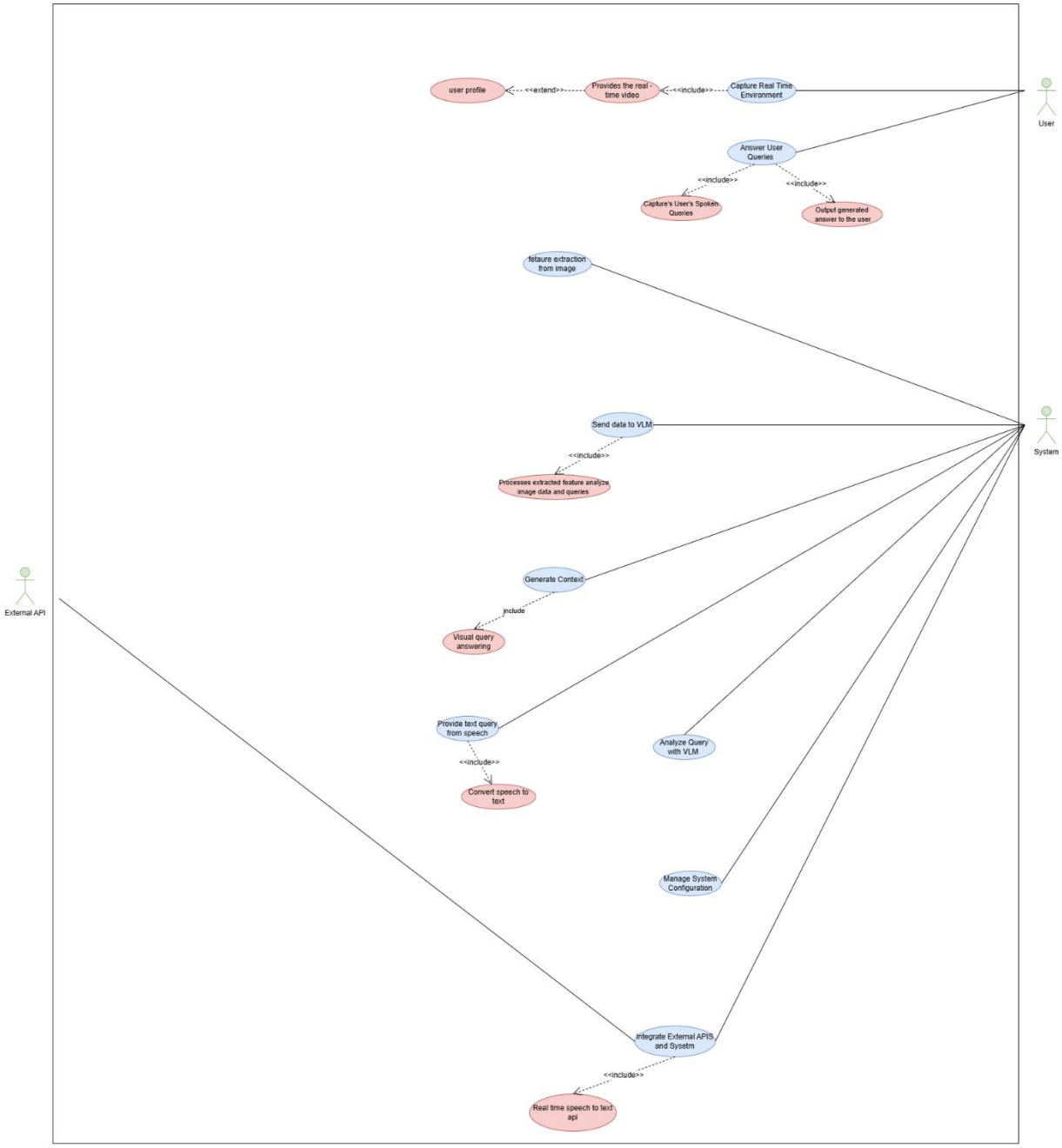


Figure 11:fig of use case diagram

3.7.2. ERD (Entity Relationship Diagram)

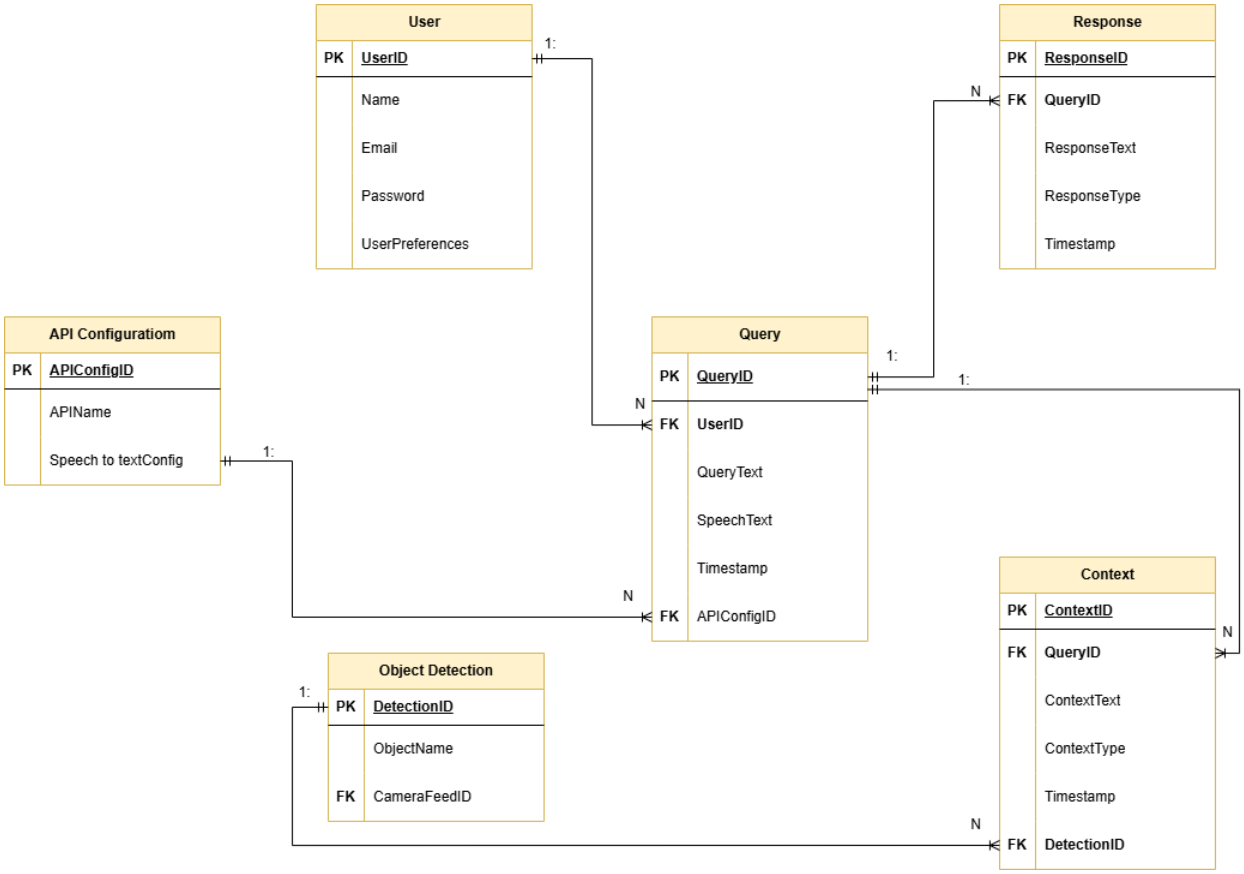


Figure 12:Figure of ERD

3.7.3. Data Flow Diagram Level 0

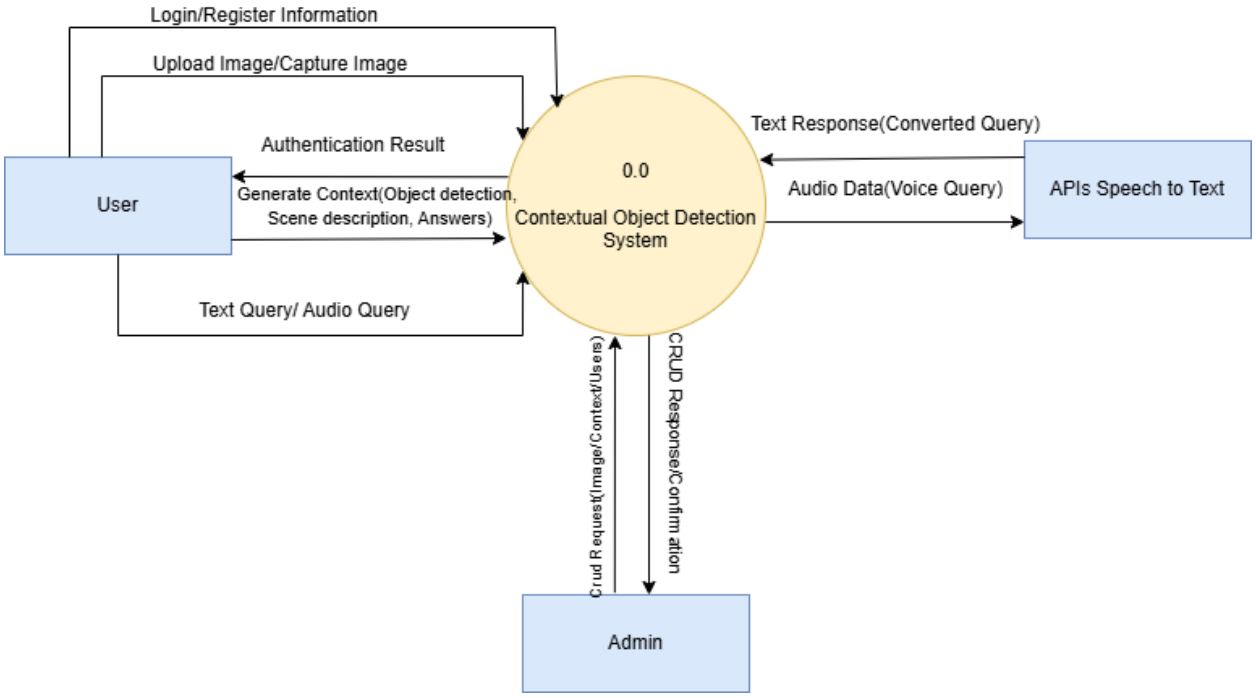


Figure 13:Figure of DFD

3.7.4. Data Flow Diagram Level 1

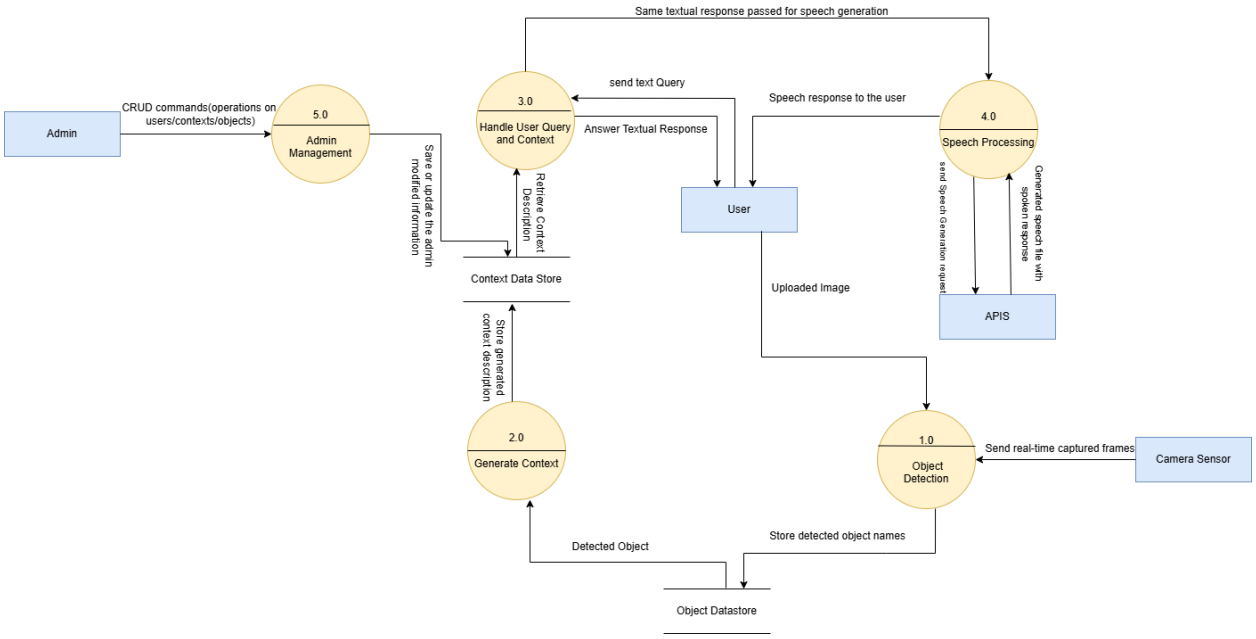


Figure 14:Level 1 DFD

3.7.5. Sequence Diagram Login

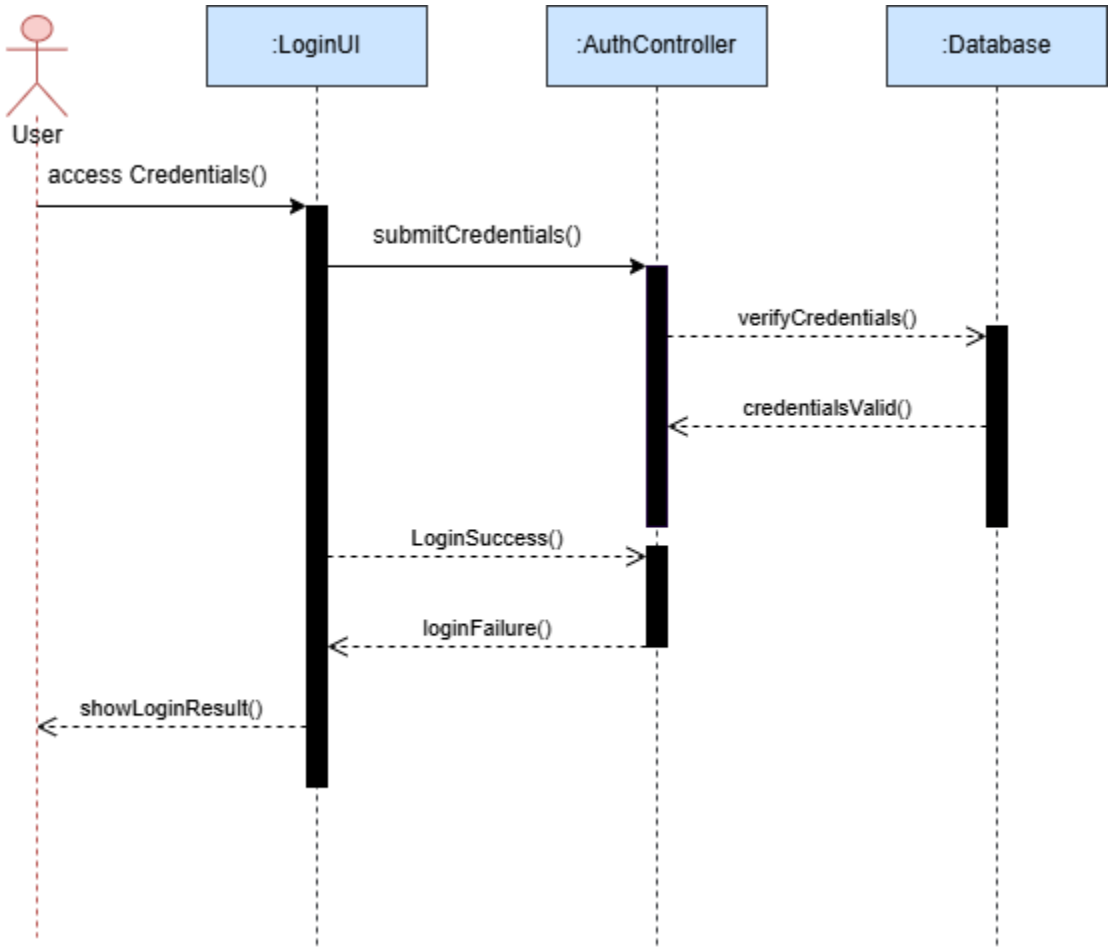


Figure 15:fig of sequence diagram Login

3.7.6. Sequence Diagram Image Processing

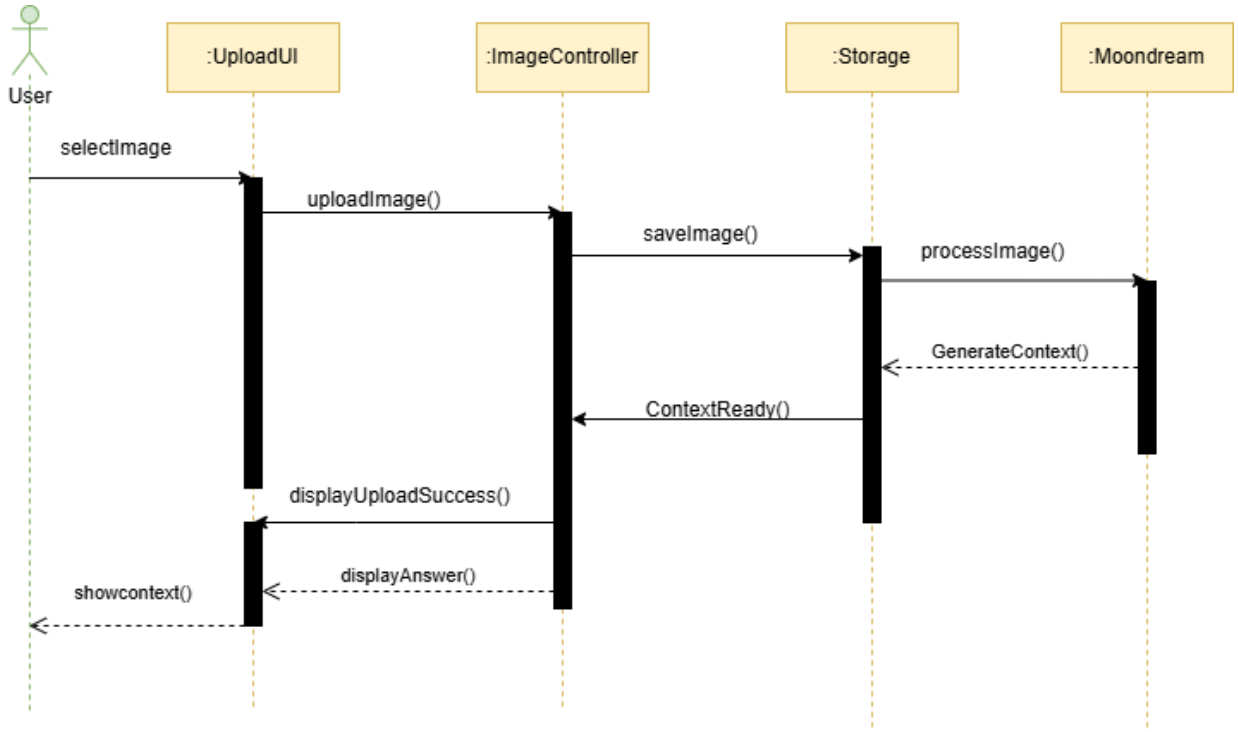


Figure 16:fig of sequence image processing

3.7.7. Sequence Diagram of System

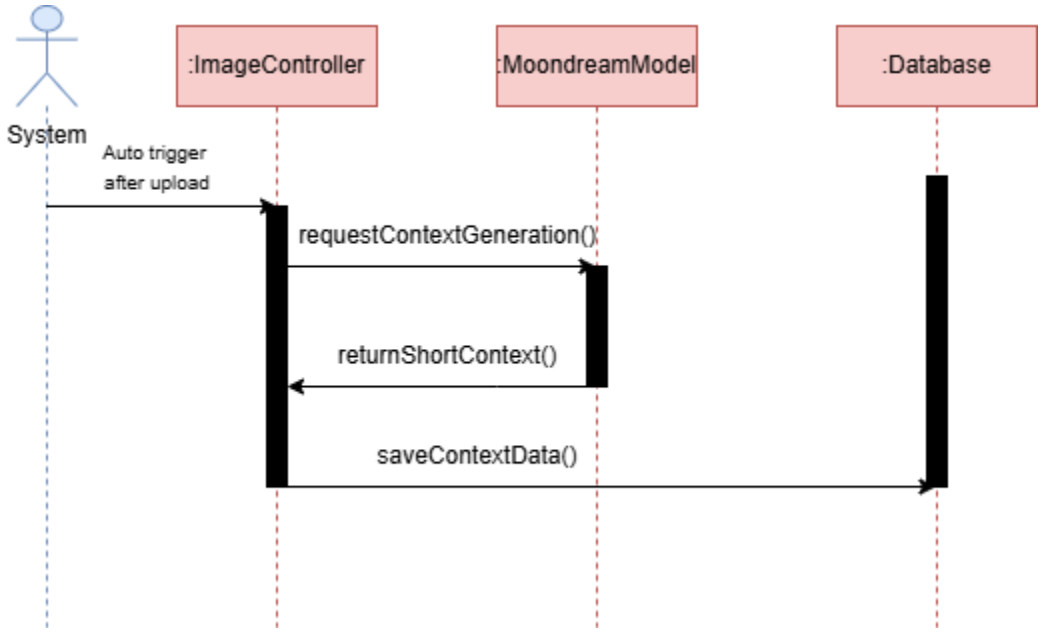


Figure 17:fig of sequence diagram of system

3.7.8. Sequence Diagram of Sequence Query

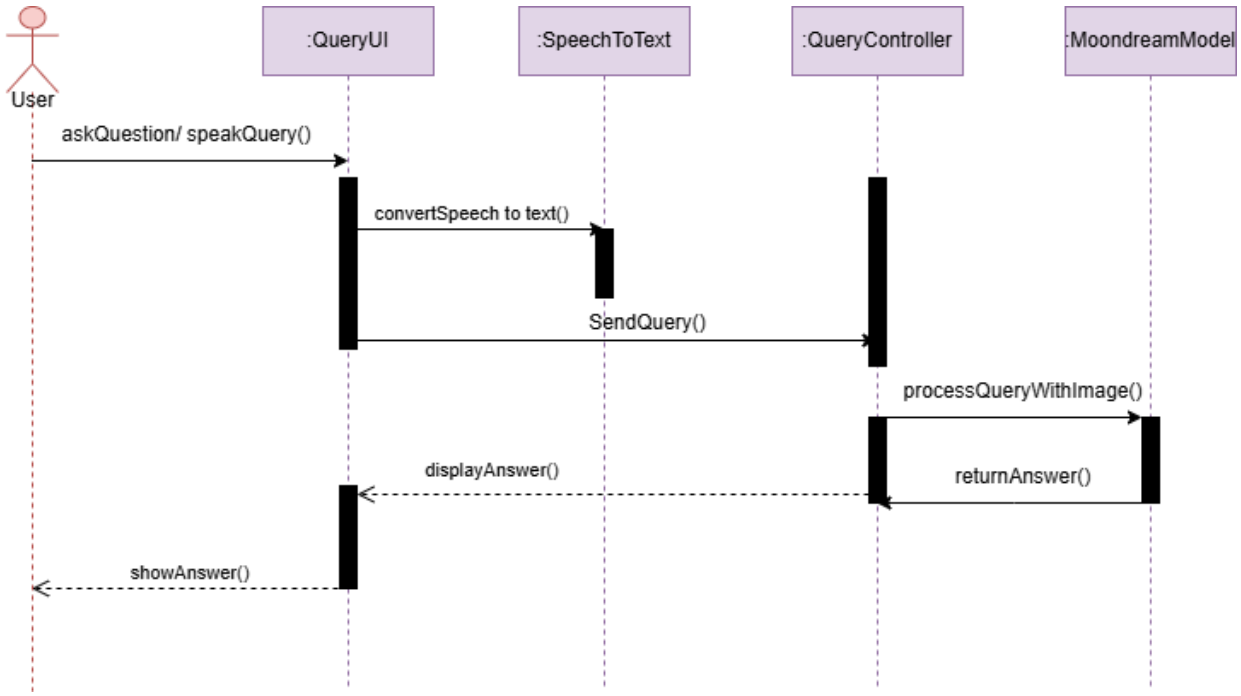


Figure 18:fig of sequence of query

3.7.9. Sequence Diagram of Admin

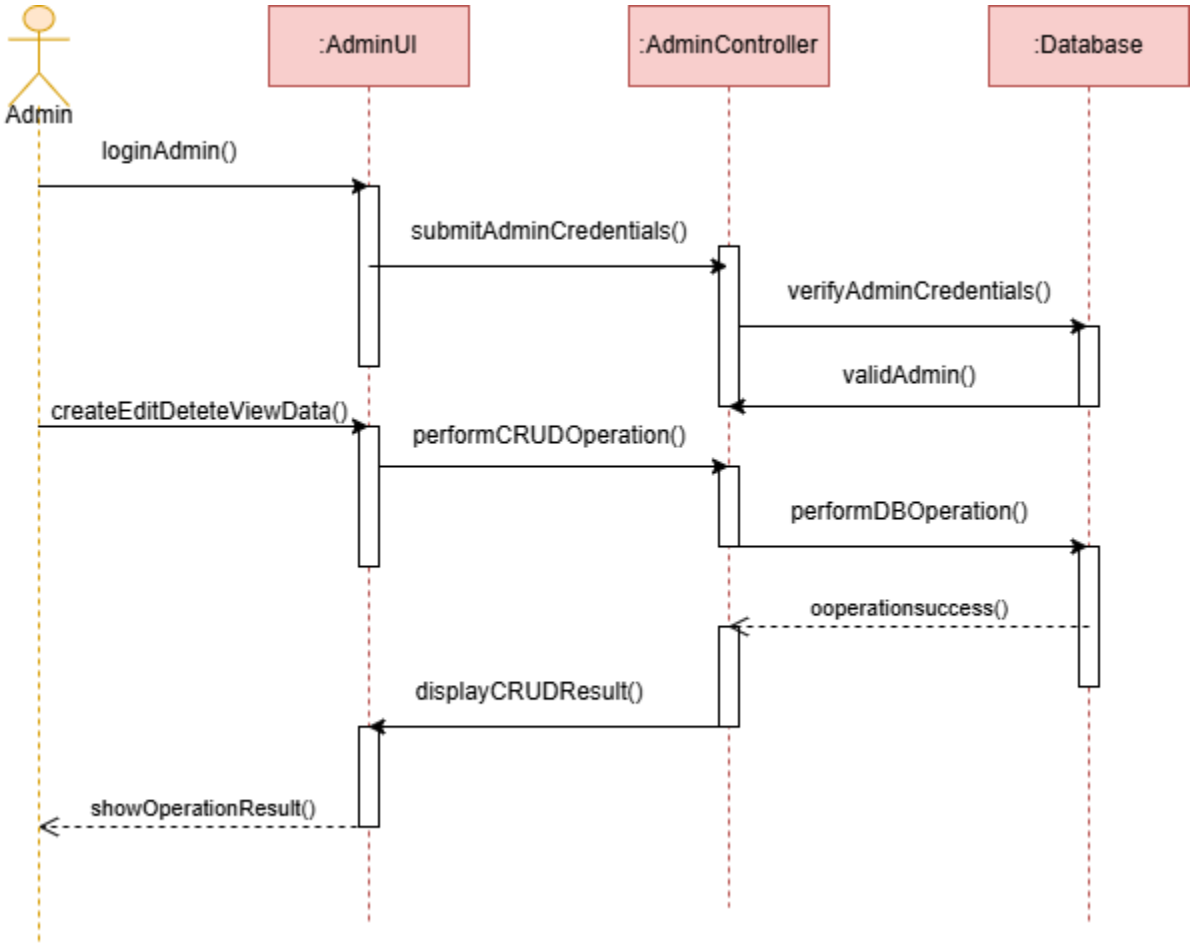


Figure 19:fig of sequence diagram of admin

3.7.10. Sequence Diagram of Profile

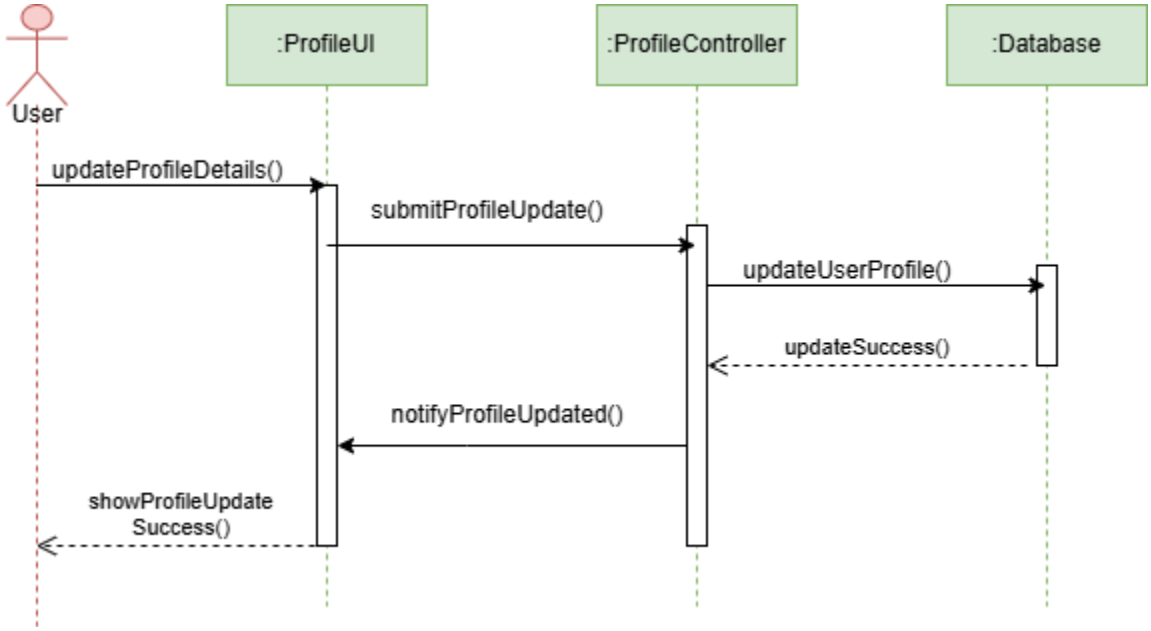


Figure 20:fig of sequence diagram of profile

3.7.11. Wireframe Login

Figure 21:fig of wireframe of login page

3.7.12. Wireframe Authentication Page

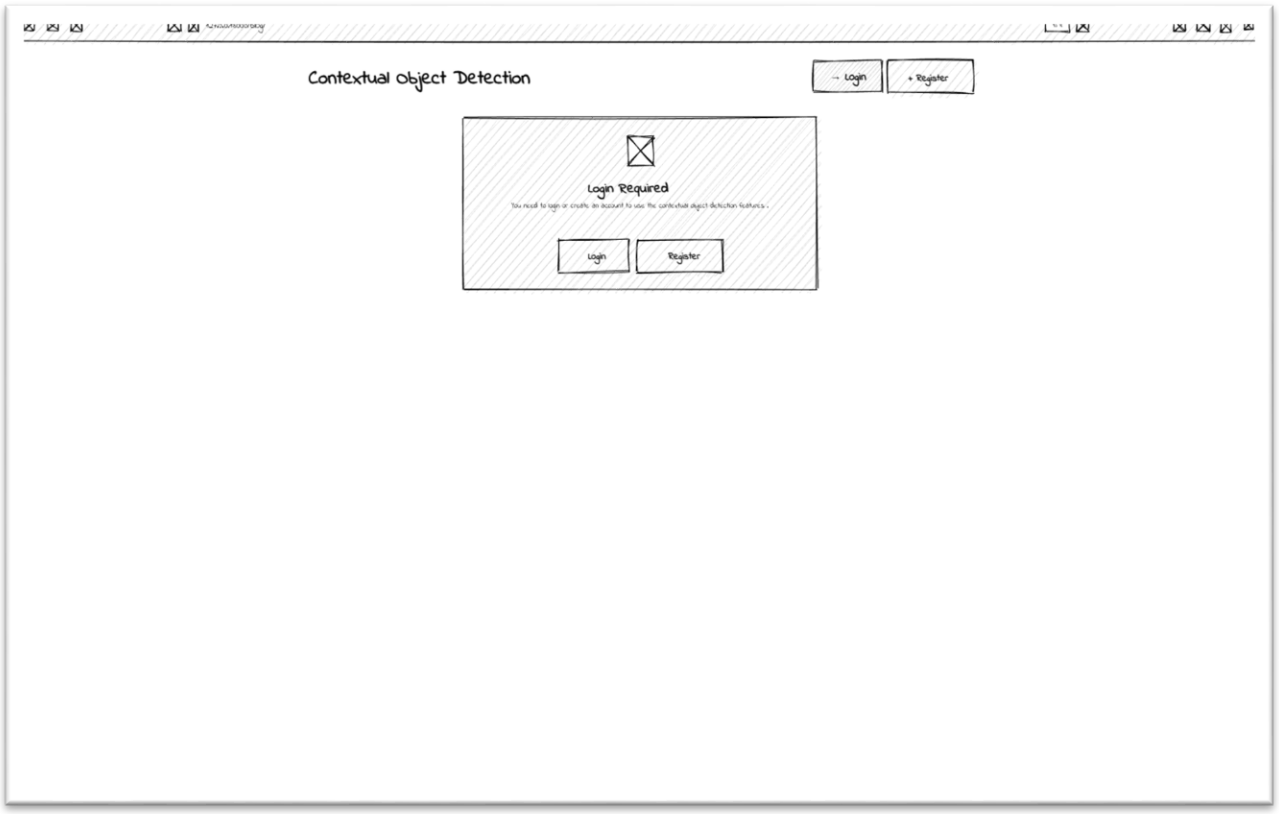


Figure 22:fig of wireframe of login and register page

3.7.13. Wireframe Register Page

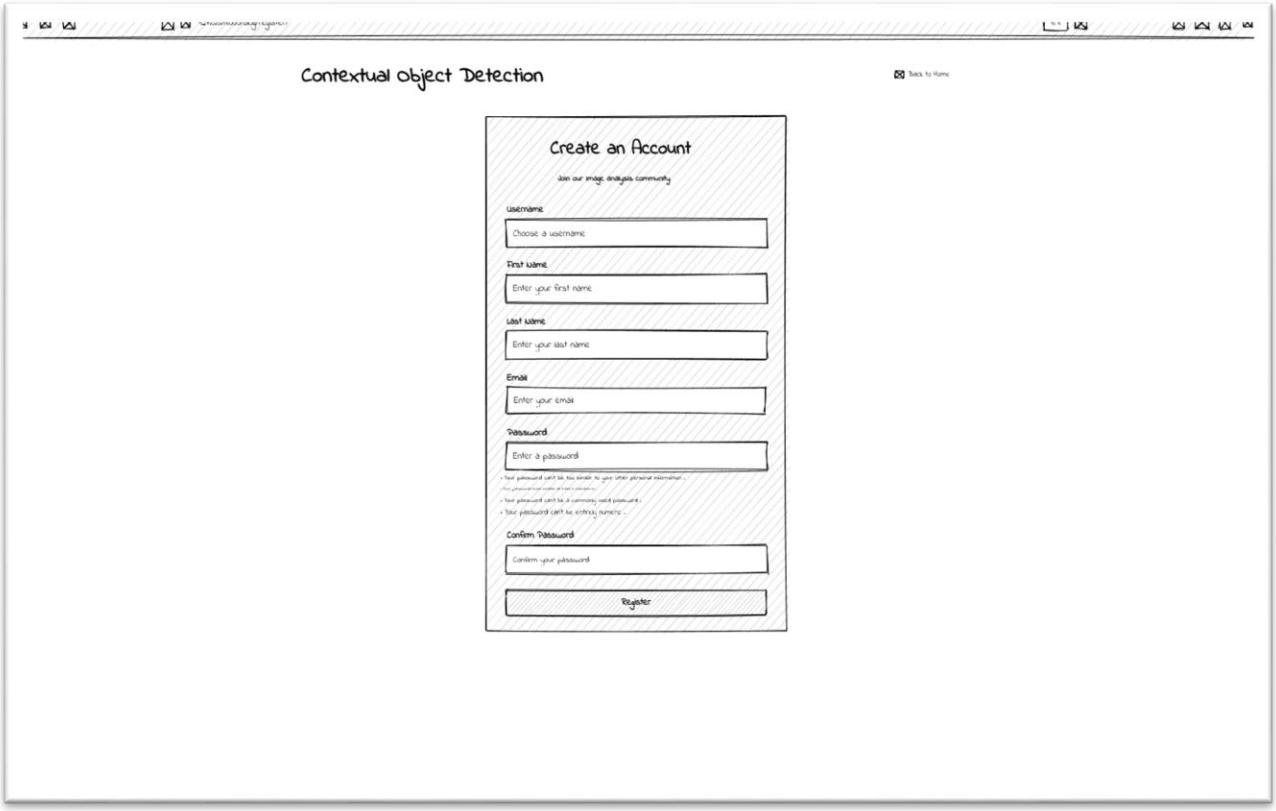


Figure 23:fig of wireframe of register

3.7.14. Wireframe of Main Page

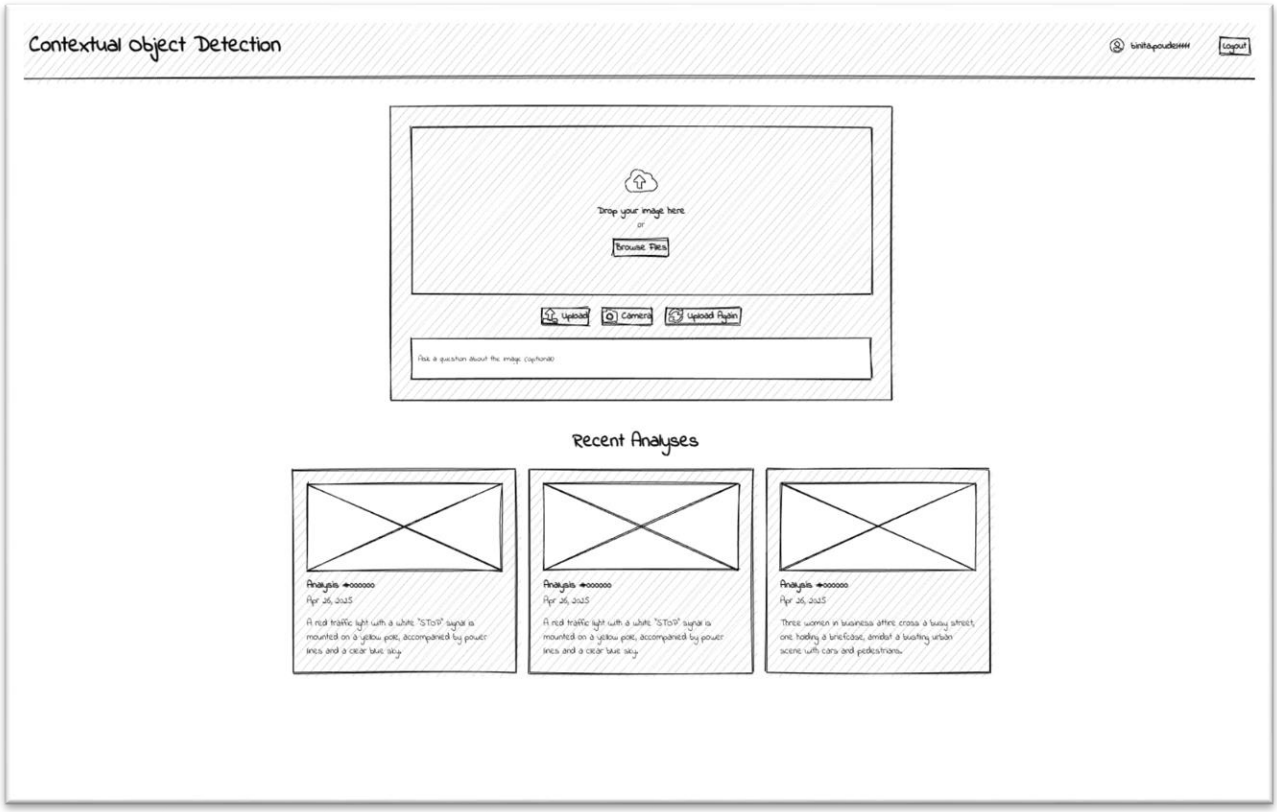


Figure 24:fig of wireframe of main page

3.7.15. Wireframe Profile Page.

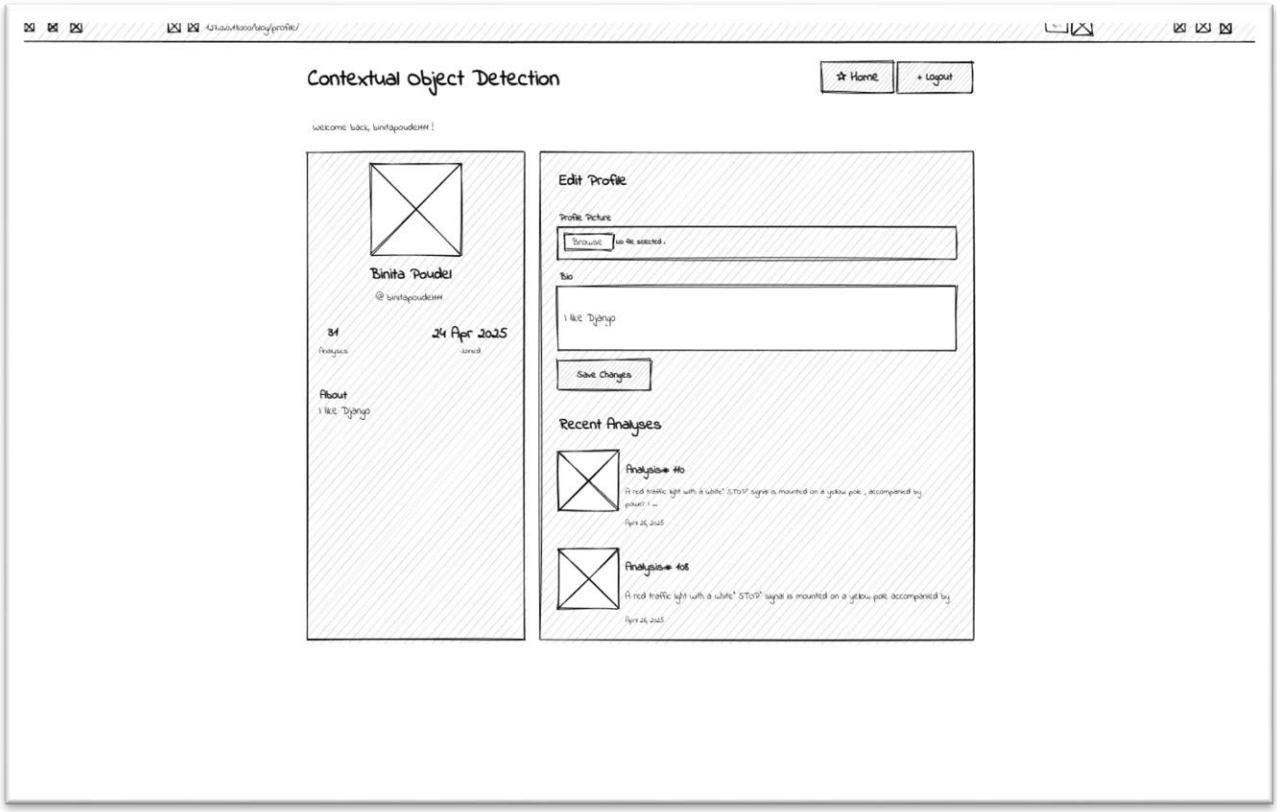


Figure 25:fig of wireframe of profile page

”

3.8. Implementation

Core Features and Code Screenshots

3.8.1. System architecture

The architecture is based on a multi-layer design on top of computer vision and natural language processing principles. The equipment uses SigLIP vision encoder and Phi-1. 5 transformer-based language model (1.86B parameters) in a hierarchical manner. User interaction is performed through a Django web application and background processing queue management for asynchronous context generation is taken care of by Redis. The Moondream2 visual language model constitutes the backbone of the integration process: context generation in real-time with low computational complexity. This design accomplishes the effective fusion of V-L processing abilities while being access friendly for resource limited conditions, which is important for assistive technology scenario.

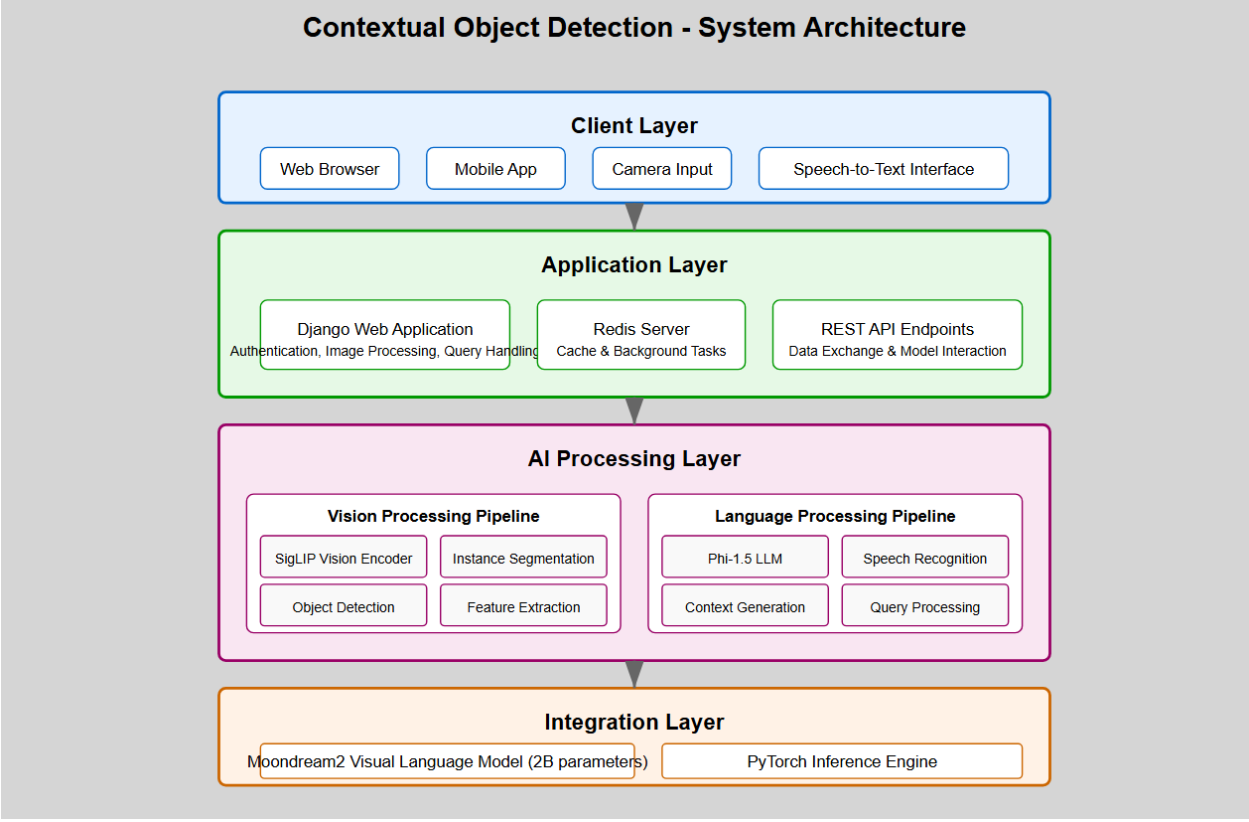


Figure 26: System Architecture for the Project

4. Chapter 4: Testing

4.1. Agile Sprint-wise Testing

Agile sprint-wise testing refers to the continuous and iterative approach to software testing that occurs during each sprint in an Agile development lifecycle (Tricentis Testim, 2019). It means that testing activities are integrated within each sprint rather than postponed to a later phase. Sprint-wise testing ensures continuous quality and faster feedback within each Agile sprint, leading to a more reliable and adaptable product (Tricentis Testim, 2019).

4.1.1. Test 1: Test for features from Sprint 3

Objective	To test the login and context generation feature from sprint 3.
Action	i. Log in to the system with credentials ii. Upload image iii. Generate context
Expected Result	The system should allow the log in with correct credentials and generate the context in seconds
Actual Result	The system allows the log in and generates the context for the image.
Conclusion	Test passed.

Table 3: Agile Test 1: To test the login and context generation feature (sprint 3)

Screenshots

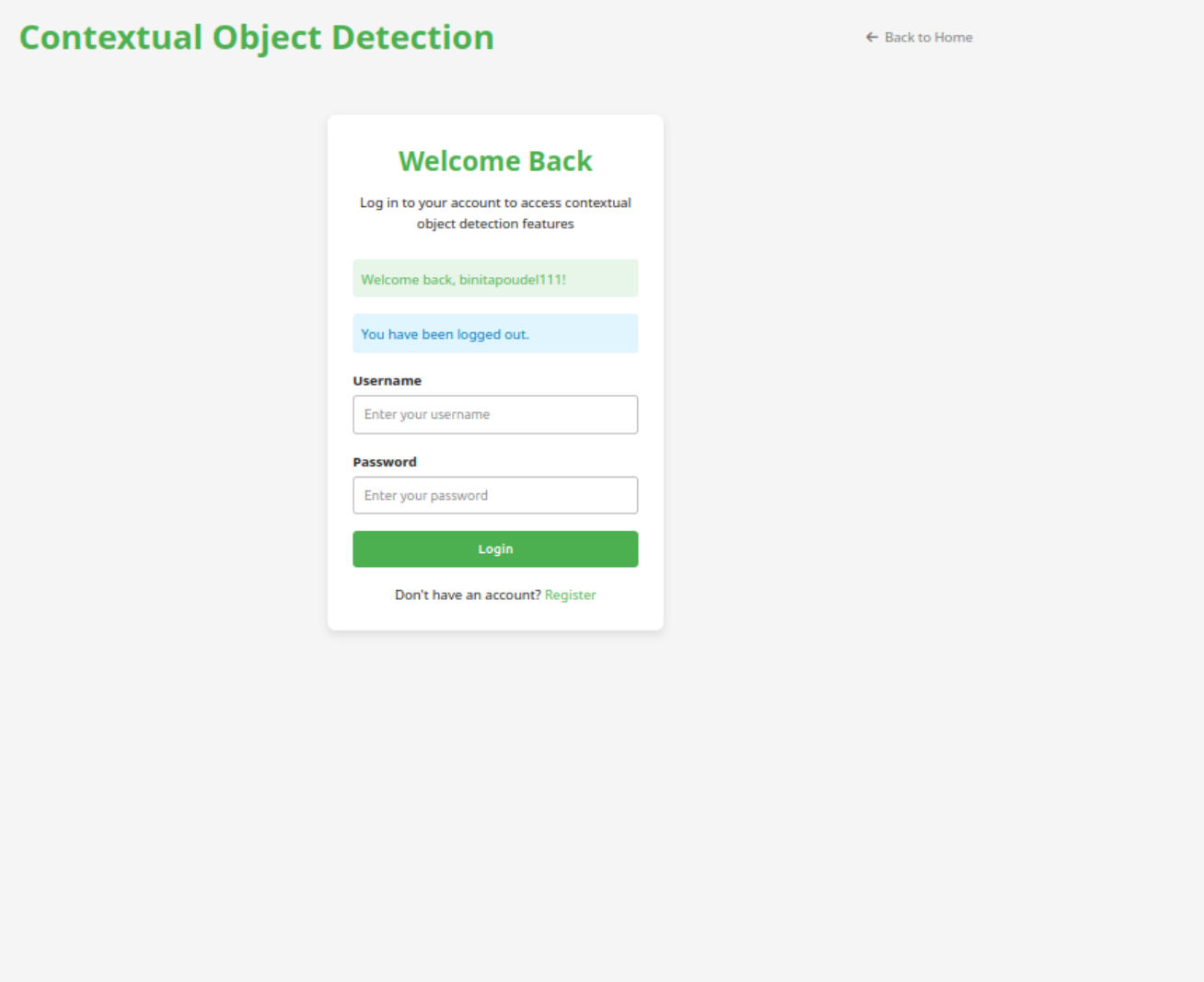


Figure 27: Login feature from sprint 3

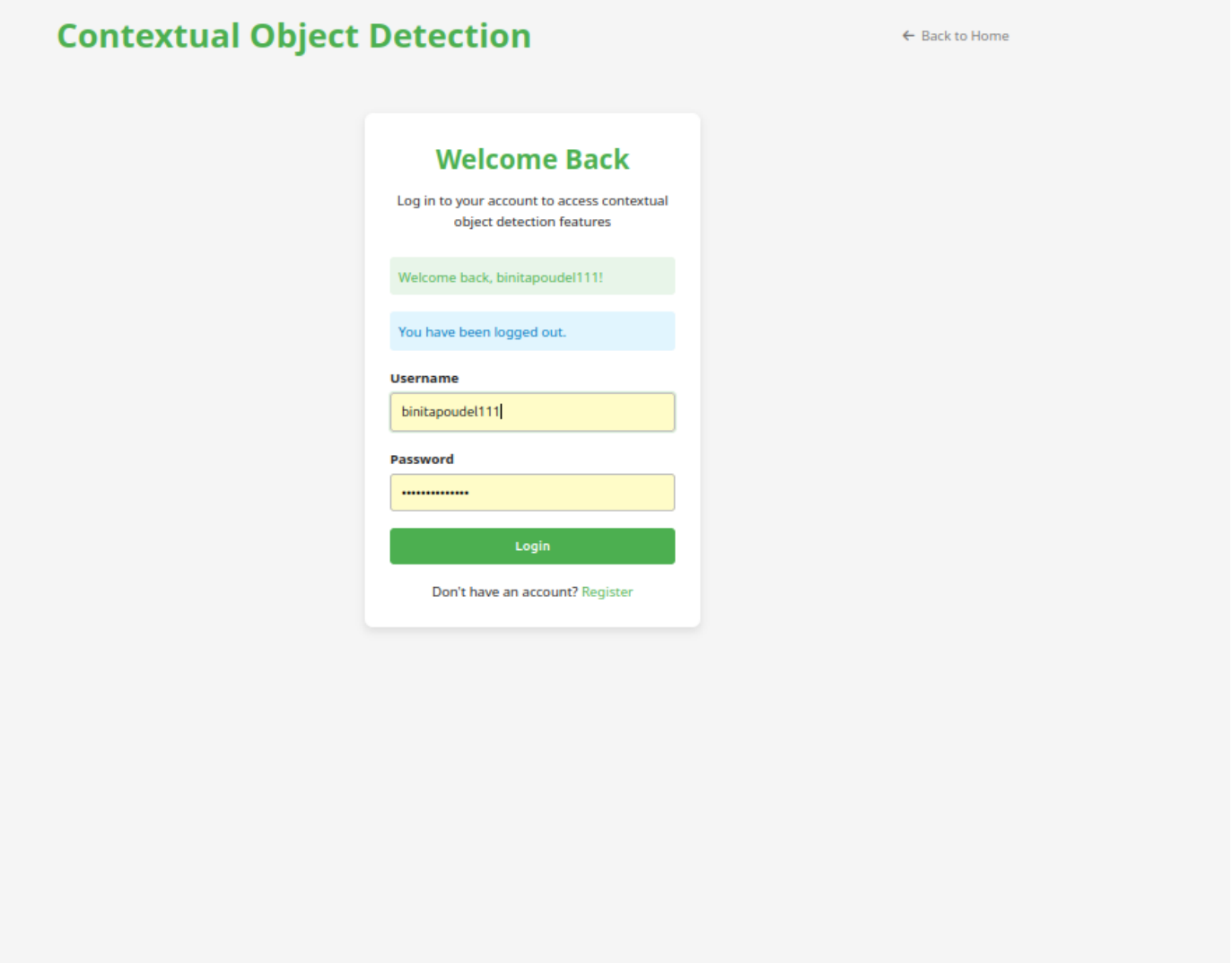


Figure 28: Working login feature

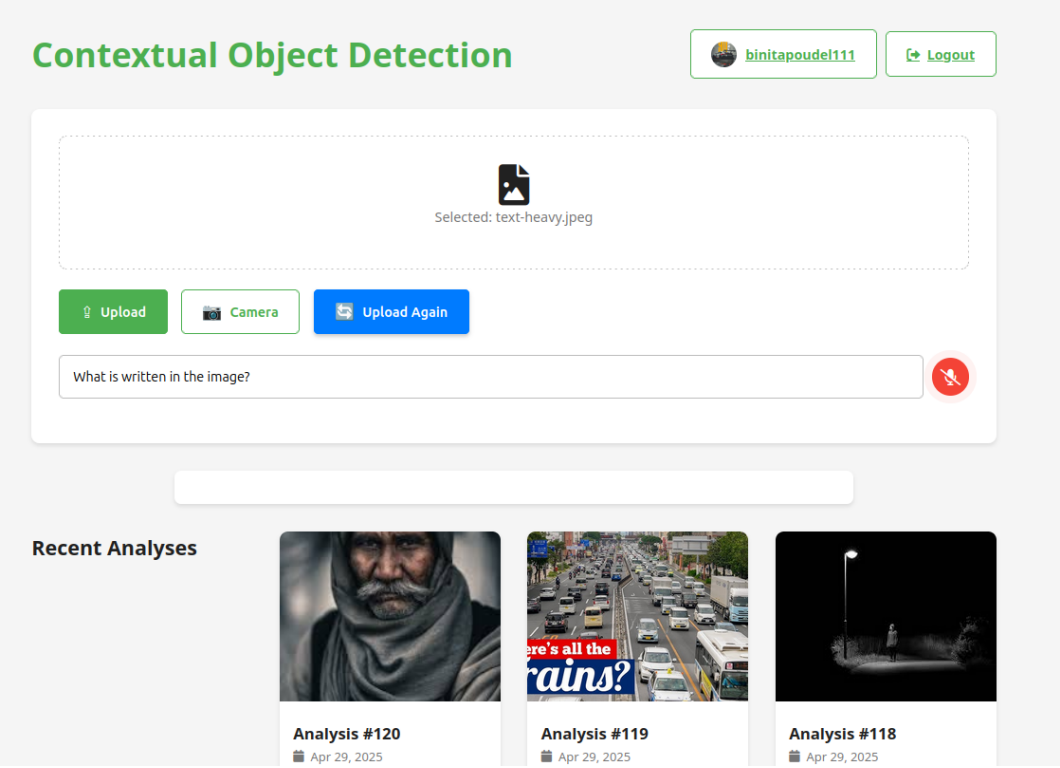


Figure 29: Greeted to the system for context generation

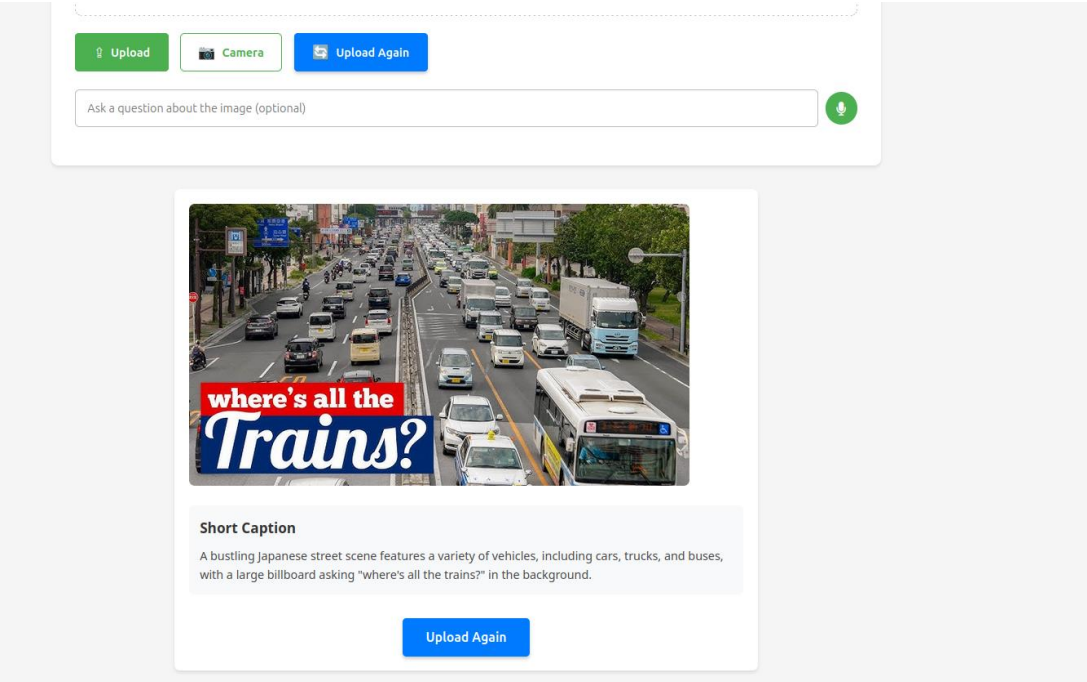


Figure 30: Successful context generation by the model from sprint 3

4.1.2. Test 2: To test the speech-to-text feature, user profile and its seamless integration in Sprint 4

Objective	To test the speech-to-text feature, user profile and its seamless integration in Sprint 4
Action	• Open context generation page. • Start microphone service • Speak the query • Generate the context • Update the profile
Expected Result	The system should successfully convert speech to text, generate context and query answer, allow the profile update.
Actual Result	The system successfully converted speech to text, generated context and query answer, allowed the profile update.
Conclusion	Test passed.

Table 4:Agile Test 2: To test the speech-to-text feature, visual query and profile update (sprint 4)

Screenshots

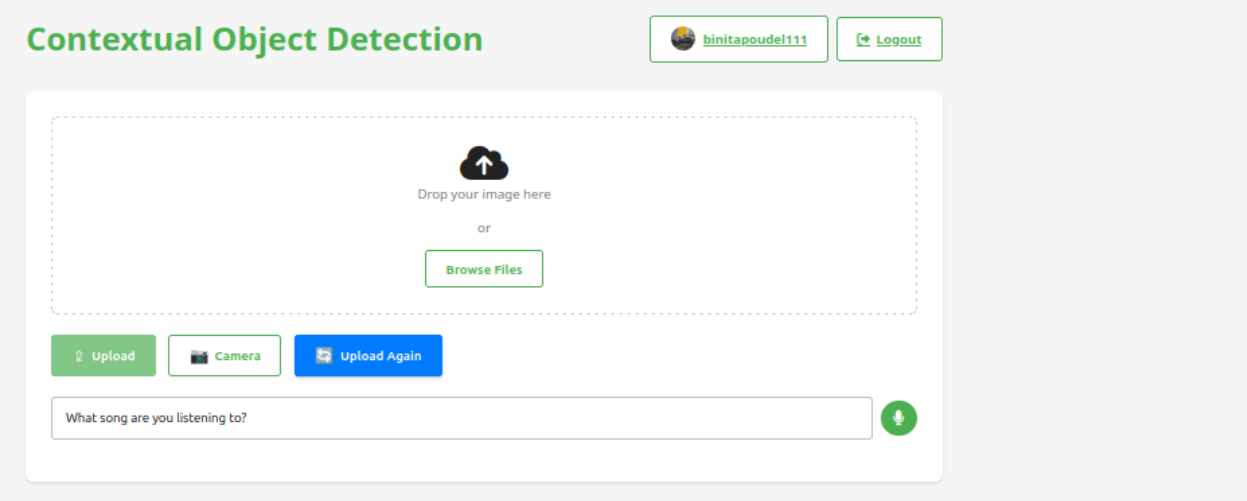


Figure 31: Enabling microphone for speecht to text

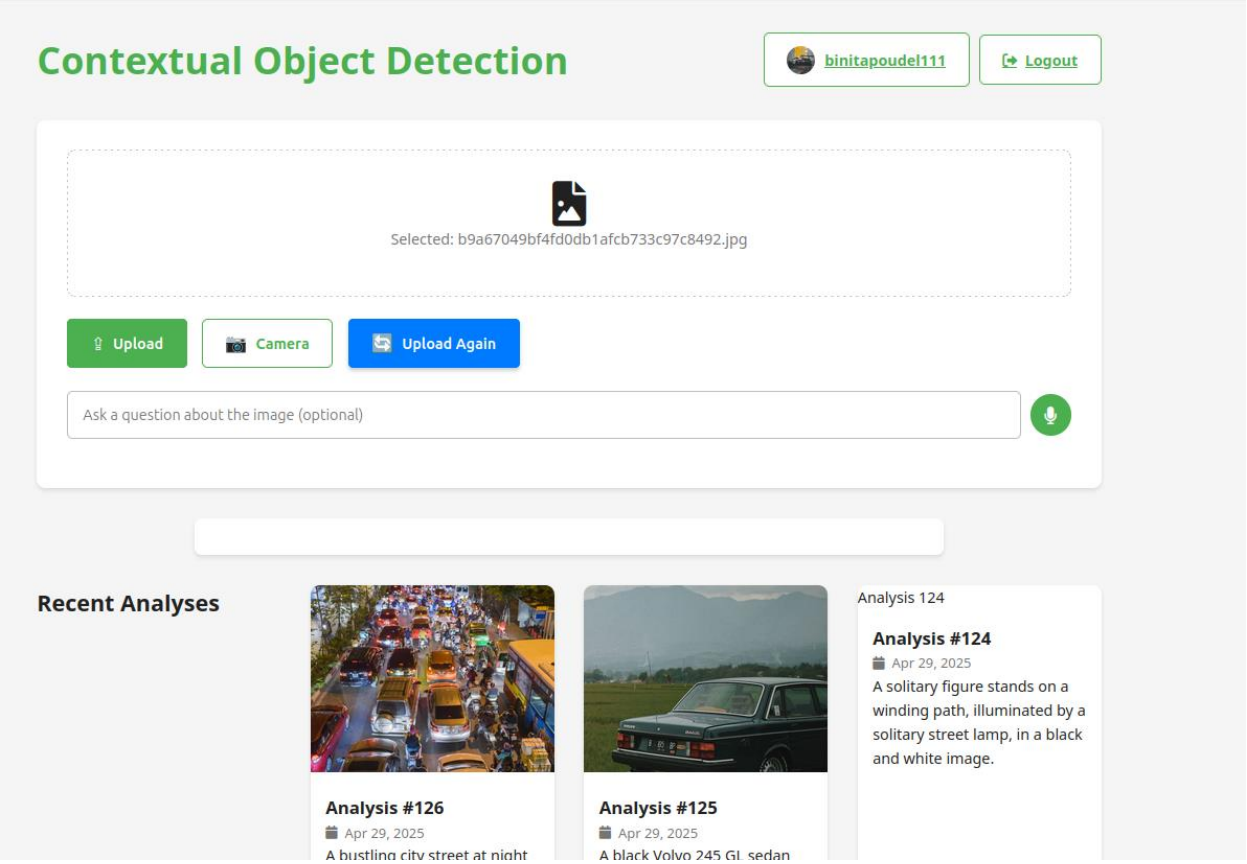


Figure 32: Uploaded image

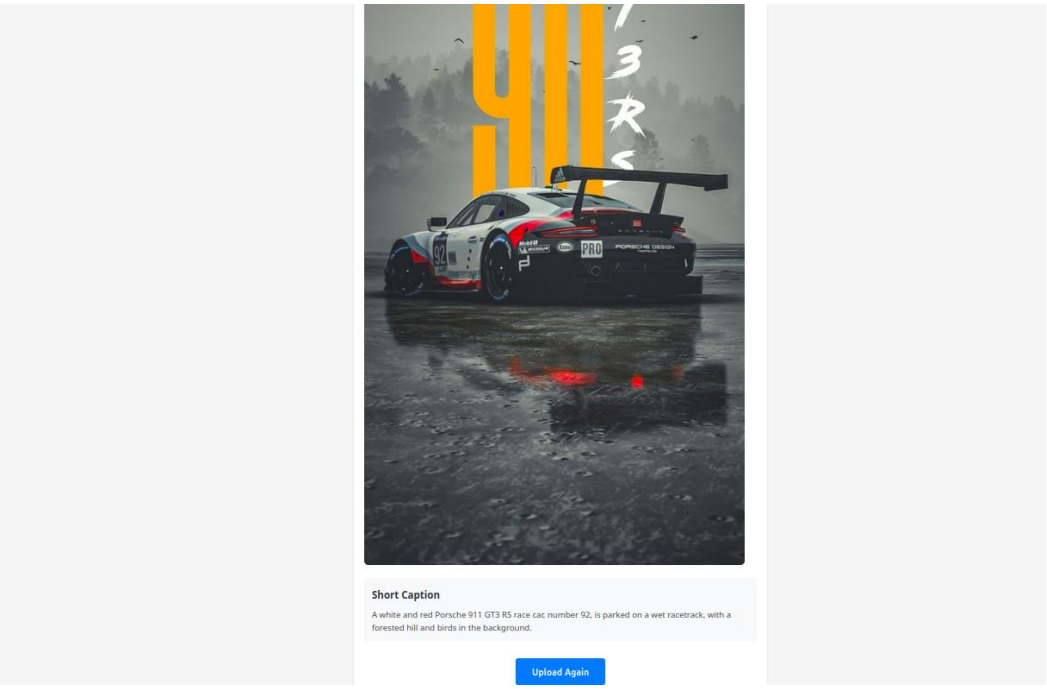


Figure 33: Context Generation with visual query and stt

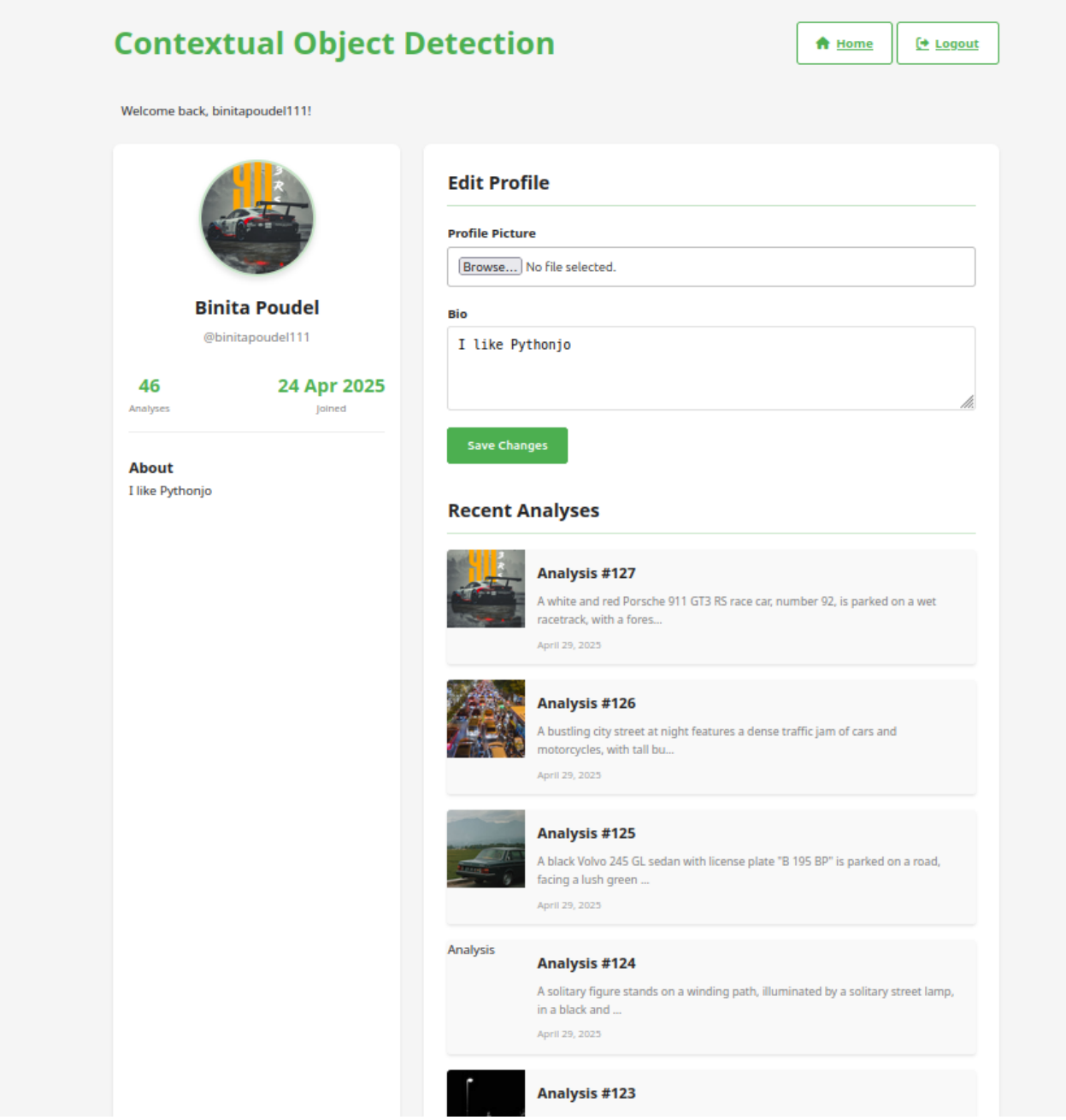


Figure 34: Updating the profile picture and bio

4.2. Unit Testing, Test Plan

The functionality and reliability of the project's components are validated through unit testing. Contextual Object Detection have different components that builds up the complete system including image upload and real time camera capture, short context generation, answering user's queries, speech-to-text, user authentication. These functionalities need to be validated before they can be deployed for the real-world users to address the corner cases that can possibly occurs in real world usage of the web application. The test also assesses the ability of the system to generate correct context for the scene along with accurate answer to the user's query along with efficient conversion of speech to text for user queries.

Test and Objectives of the test are further listed in the given table.

Test	Objectives
1	User Registration
2	Login for registered users
3	Verify non-registered user cannot login
4	Correct login and redirection to main page
5	Logout and redirection to authentication page
6	Confirm user can update their profile
7	Check user's access to their profile
8	Confirm user can add bio to their profile
9	Check whether the profile picture shown in the context generation page.
10	Capture image in real time and process it for context generation without visual query
11	Capture the image in real time and process it with visual query for context generation.
12	Check browsing the image functionality works
13	Confirm that the upload button works and context generation starts
14	Consecutive image upload for context generation
15	Invalid Image Upload
16	Corrupt image upload

17	Speech to text functionality for user query check
18	Visual query for image without speech to text functionality
19	Short context generation for the image and time taken
20	Short caption and visual query answering for the image and time taken
21	Termination of context generation
22	CRUD Operations on users' data in admin dashboard
23	CRUD operations on image analysed in admin dashboard.
24	Consecutive image upload with visual query for context generation.

Table 5: Unit testing plan

Unit Testing

4.2.1. Test Case 1: User registration with intended credentials to all fields.

Objectives	To test the user registration functionality with intended credentials to all fields
Action	User fill the information relevant to the registration credentials along with valid email and strong password.
Expected Result	User should be successfully registered.
Actual Result	User registered successfully.
Conclusion	Test passed.

Table: Test 1: User registration with inteded credentials functionality to all fields.

Screenshots

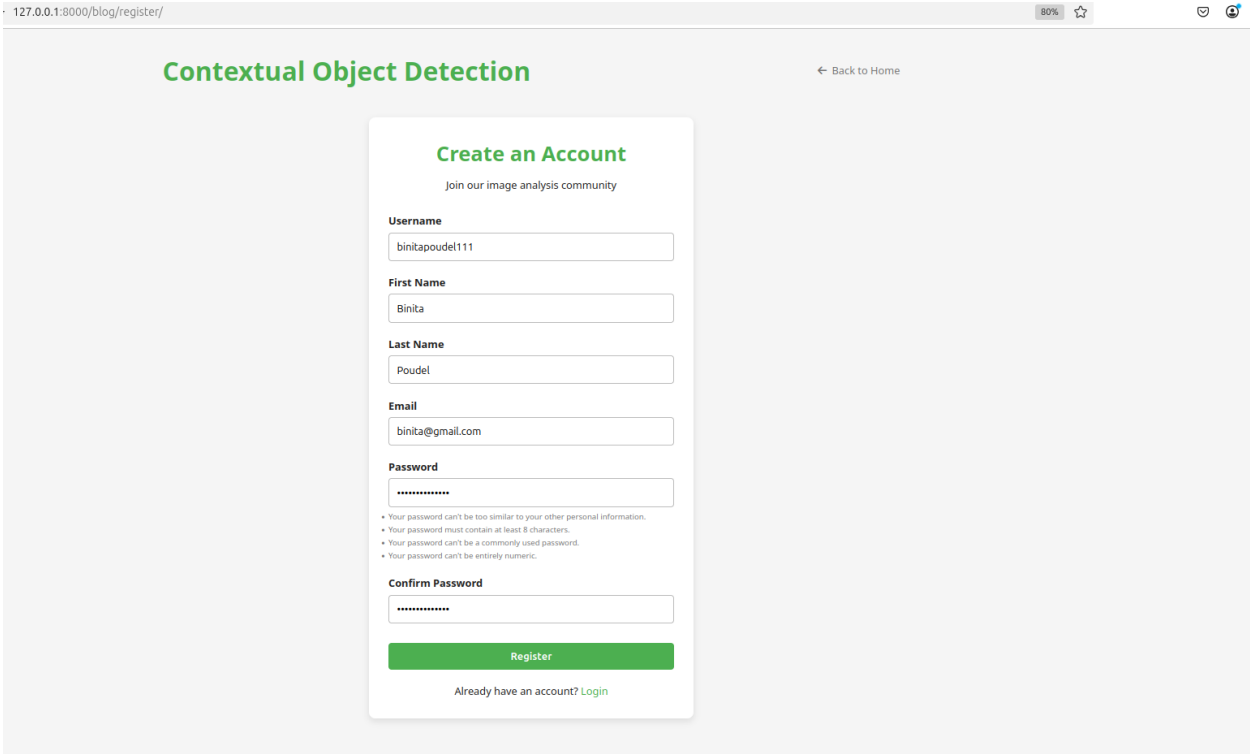


Figure 35:Registration of new user

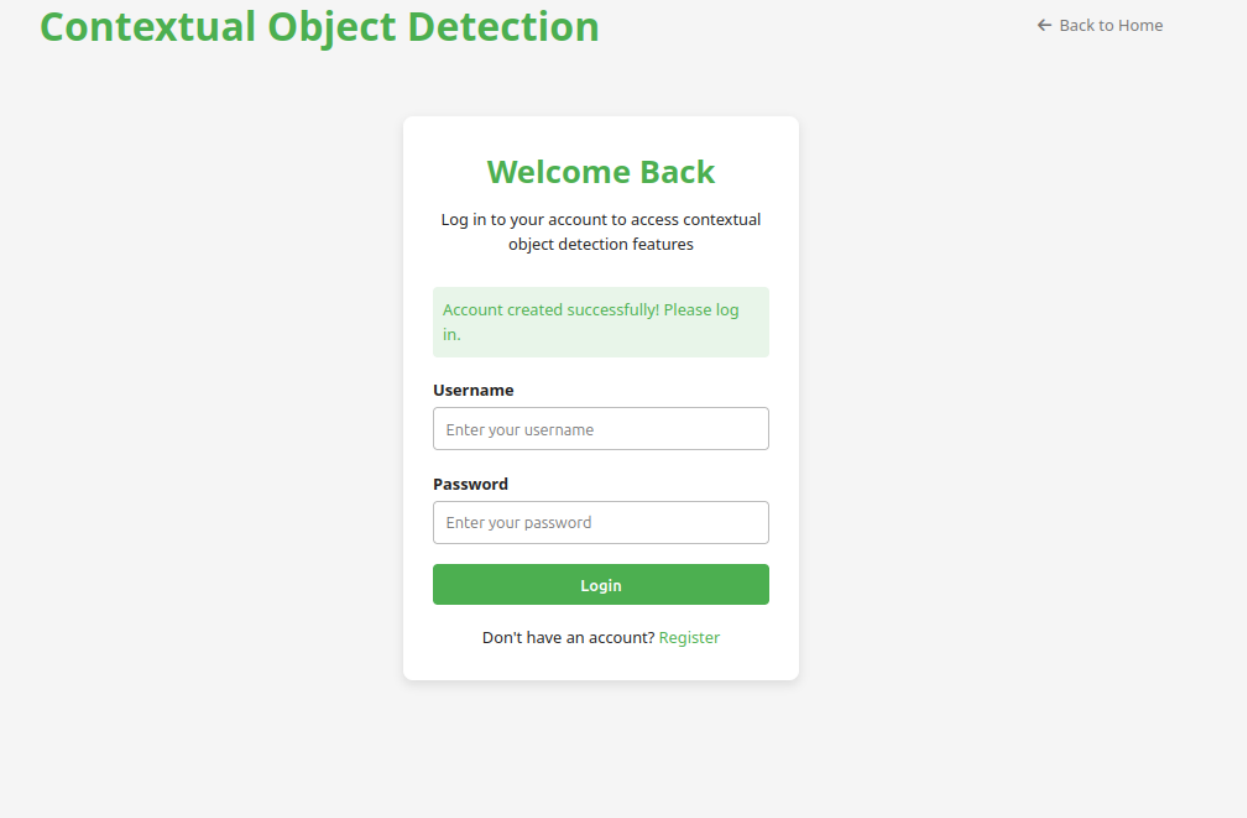


Figure 36: User registered

4.2.2. Test Case 2: To test login functionality with registered user's credentials.

Objectives	To test login functionality with registered user's credentials
Action	The username and password of the registered user is entered
Expected Result	The user should be able to login to the system.
Actual Result	User logged in successfully.
Conclusion	Test passed.

Table 6: Test 2: To test login functionality with registered user's credentials

Screenshots

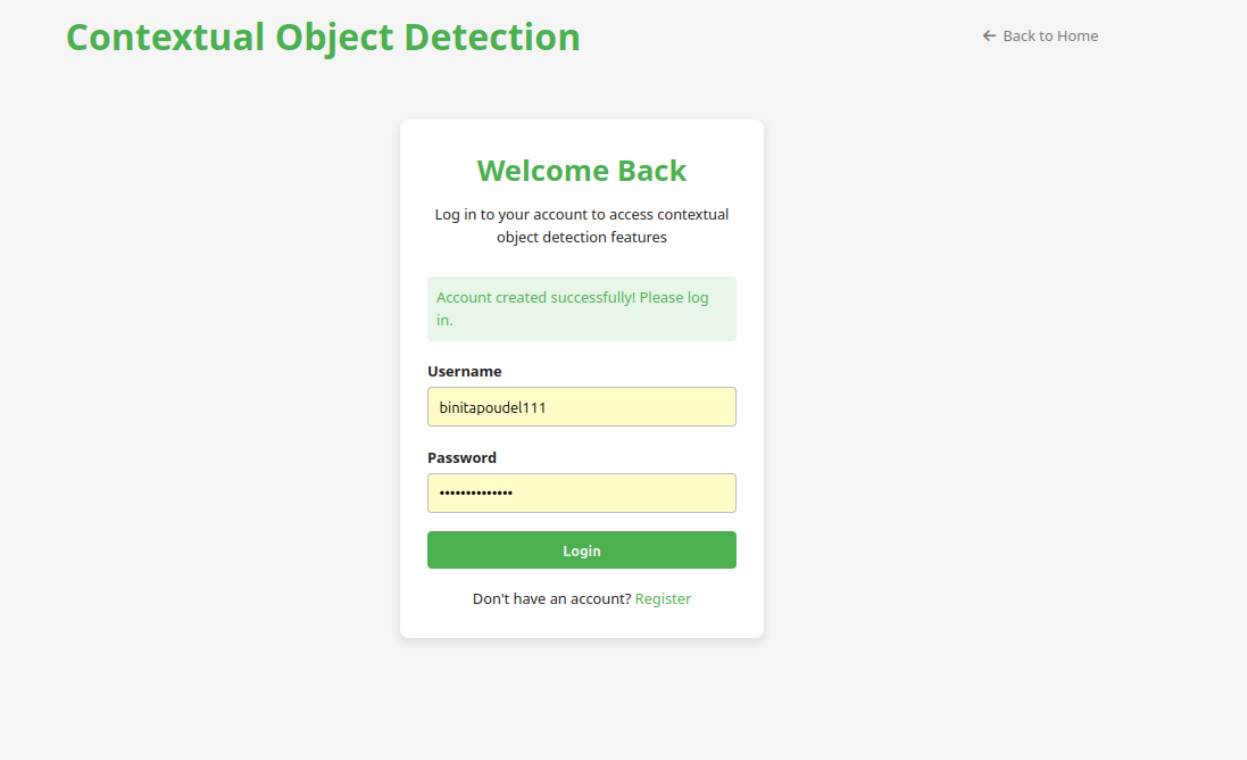


Figure 37: Filling correct credentials for login

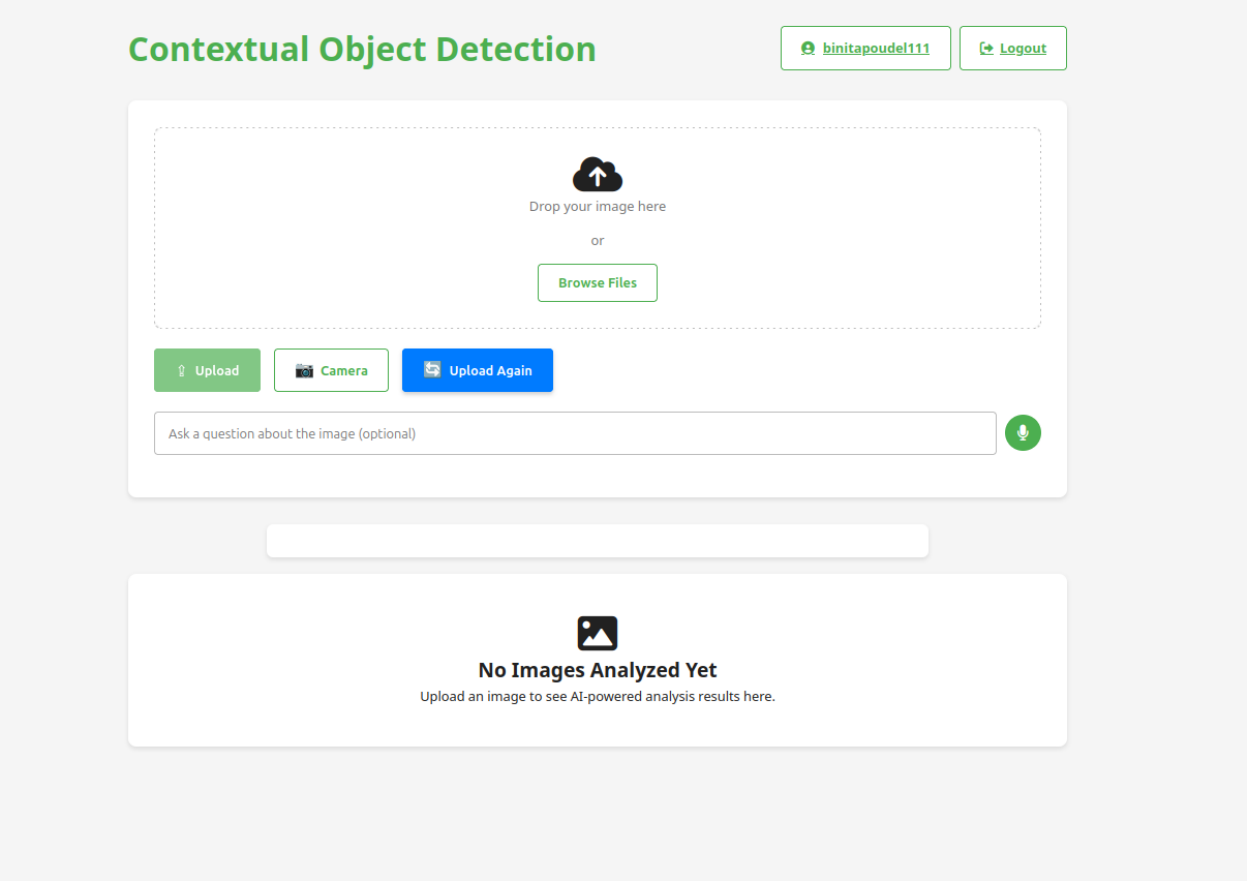


Figure 38: Main page after login

4.2.3. Test Case 3: Verify non-registered user cannot login

Objectives	To verify the non-registered user cannot login to the system
Action	Attempt to login to the system with non-registered credentials
Expected Result	The login should not be successful.
Actual Result	Login wasn't successful
Conclusion	Test passed.

Table 7: Test 3: Verify non-registered user cannot login

Screenshots

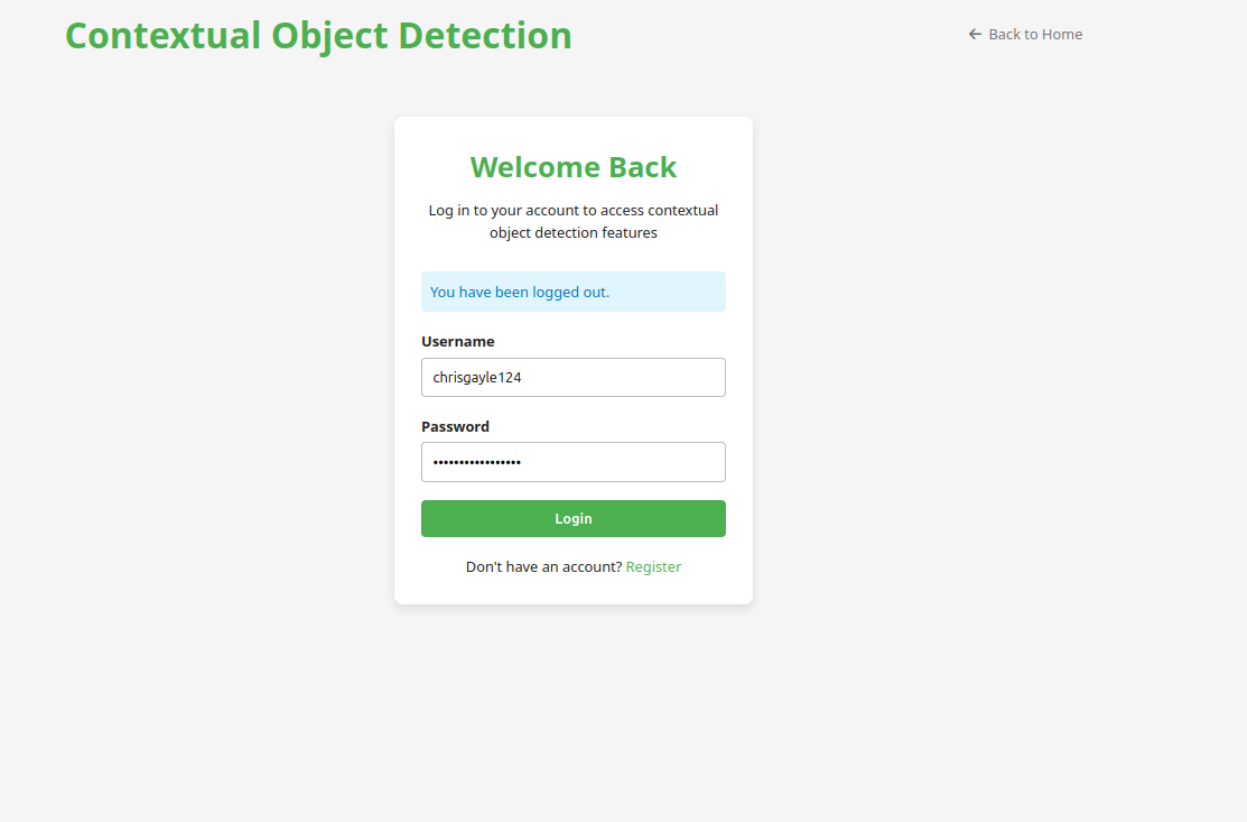


Figure 39: Login with non-registered user

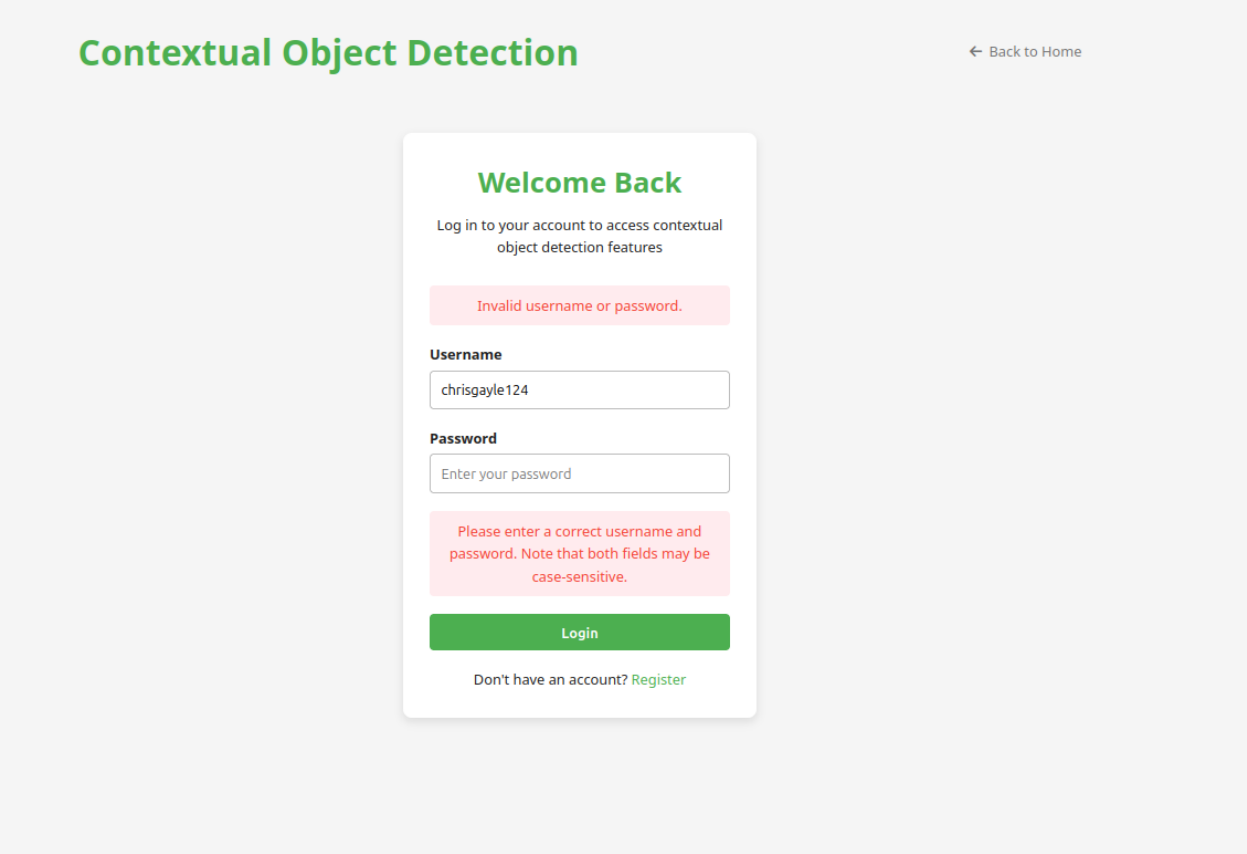


Figure 40: Login failed with unregistered user

4.2.4. Test Case 4: Correct login and redirection to main page

Objectives	To test the login with correct credentials and redirection to the context generation page.
Action	Attempt to login with registered user's credentials.
Expected Result	The user should log in successfully and redirected to context page.
Actual Result	The user logged in and redirected to main page.
Conclusion	Test passed.

Table 8: Test 4: Correct login and redirection to main page

Screenshots

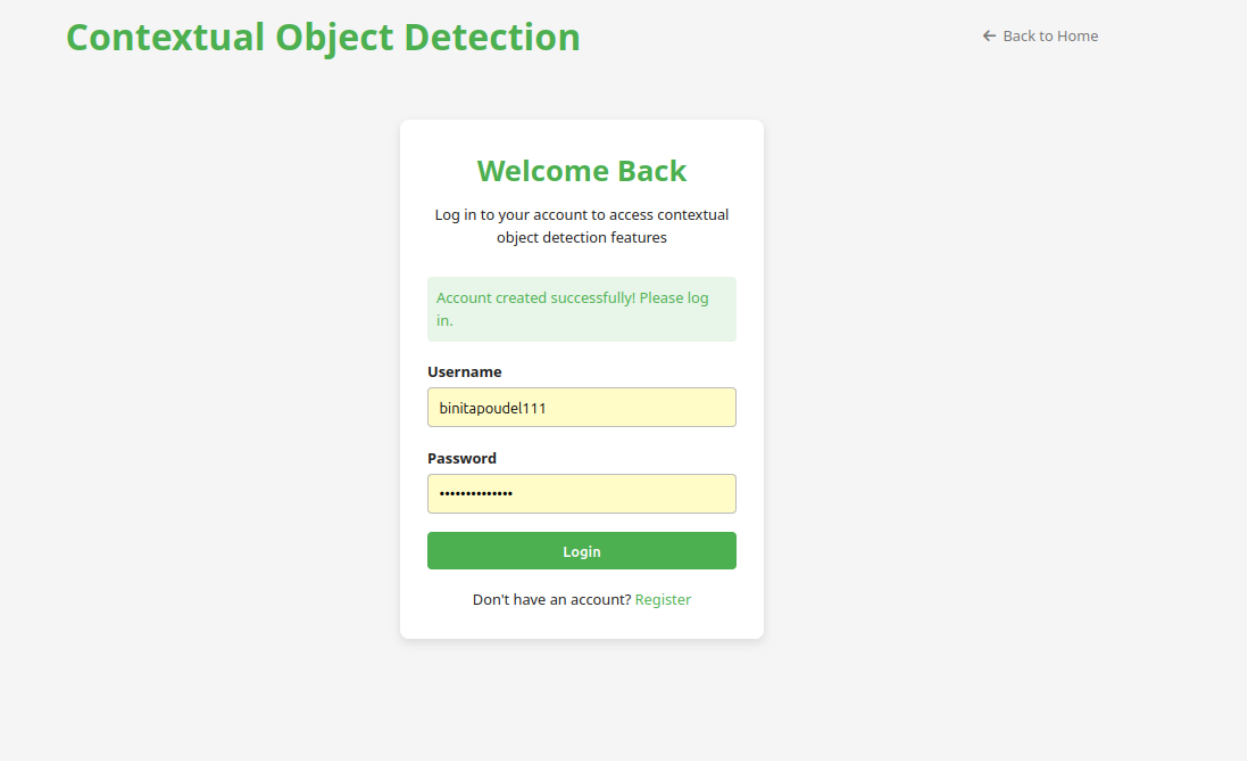


Figure 41: Login with registered account credentials

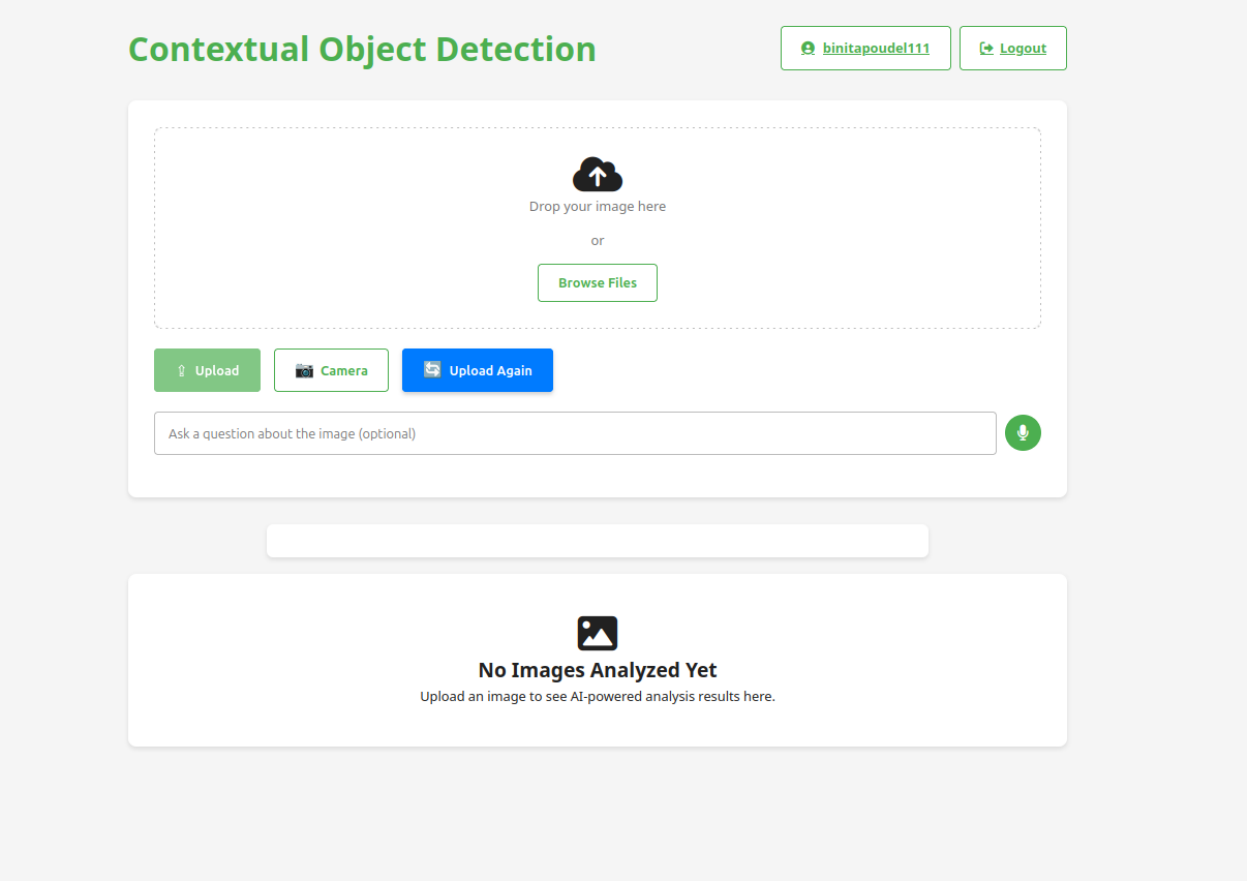


Figure 42: Successful redirection

4.2.5. Test Case 5: Logout and redirection to the authentication page

Objectives	To test the log out functionality and redirection to the authentication page.
Action	Clicked on logout button by the logged in user.
Expected Result	After the button is clicked it should logout the user and redirect to the authentication page.
Actual Result	Log out and redirection was successful.
Conclusion	Test passed.

Table 9: Test 5: Logout and redirection to the authentication page

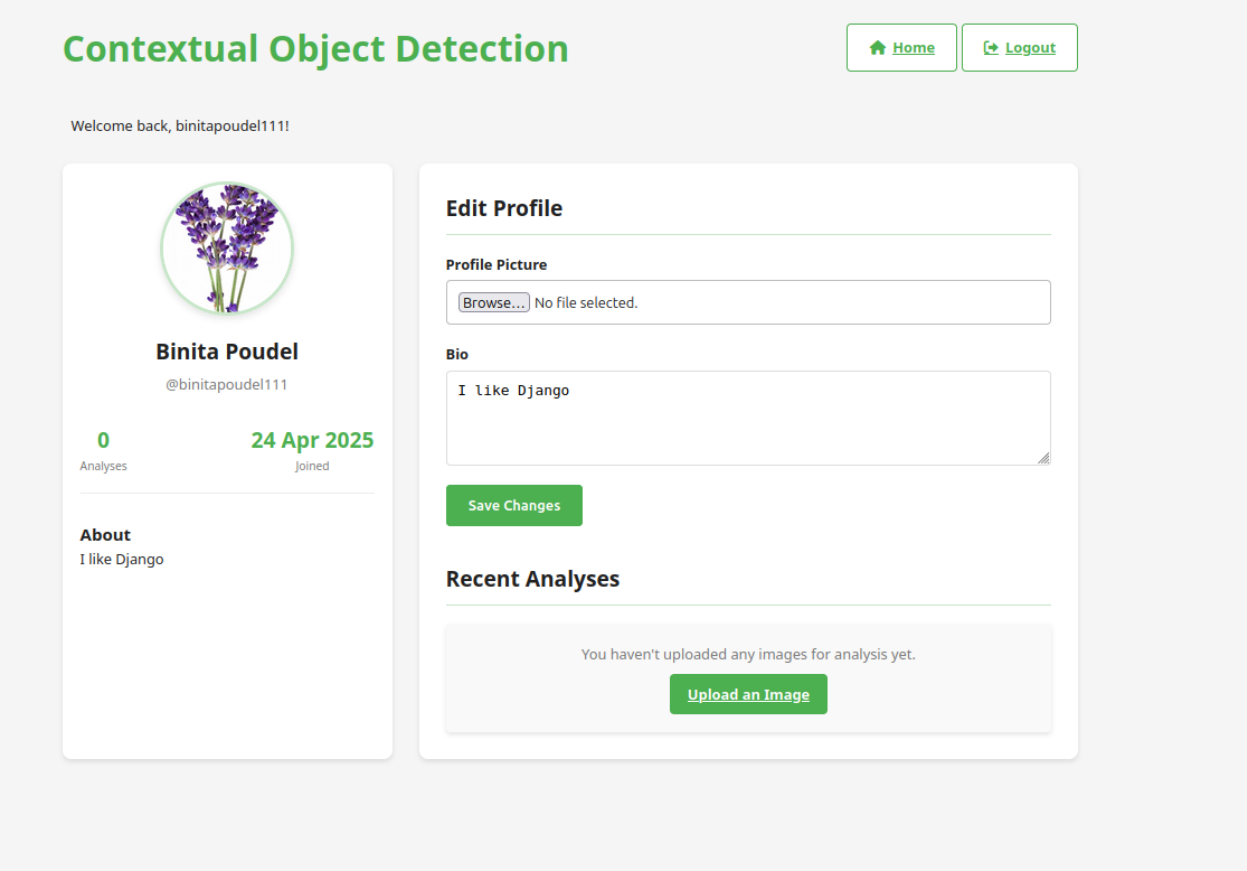


Figure 43: Logout from logged in user

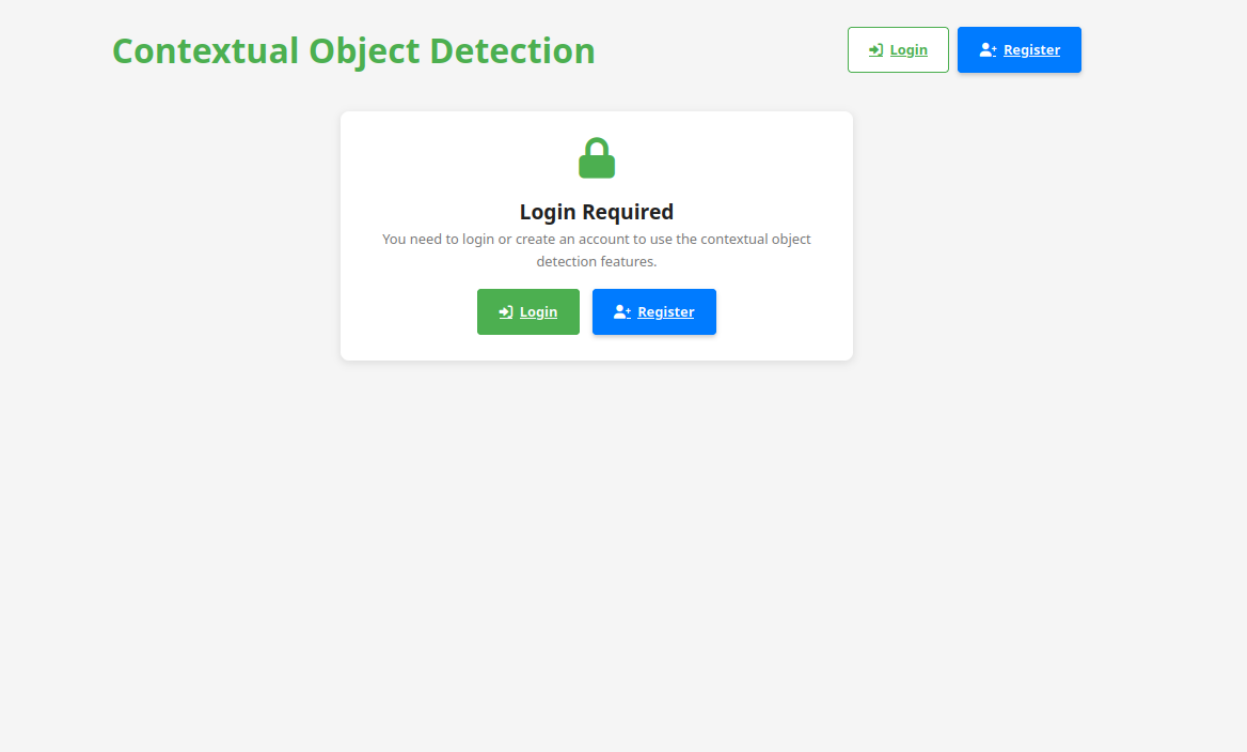


Figure 44: Redirection to authentication page after logout

4.2.6. Test Case 6: Check user's access to their profile

Objectives	To test whether the user can access their profile
Action	User clicked on the profile button in the main page for accessing their profile
Expected Result	Profile page should open with the user's information and functionality to add profile picture and adding the bio.
Actual Result	Successfully accessed the profile of the user.
Conclusion	Test passed.

Table 10: Test 6: Check user's access to their profile picture

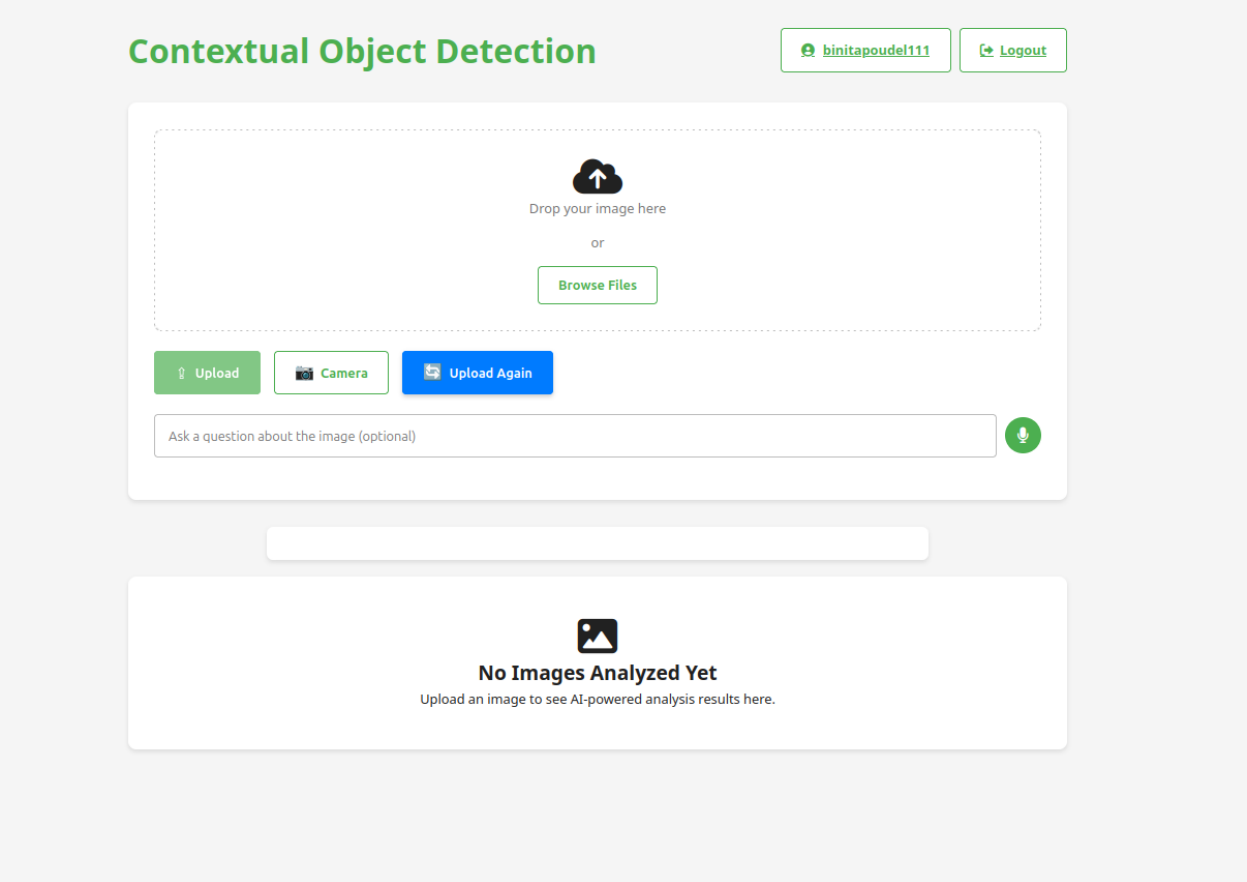


Figure 45: Main page after use log in

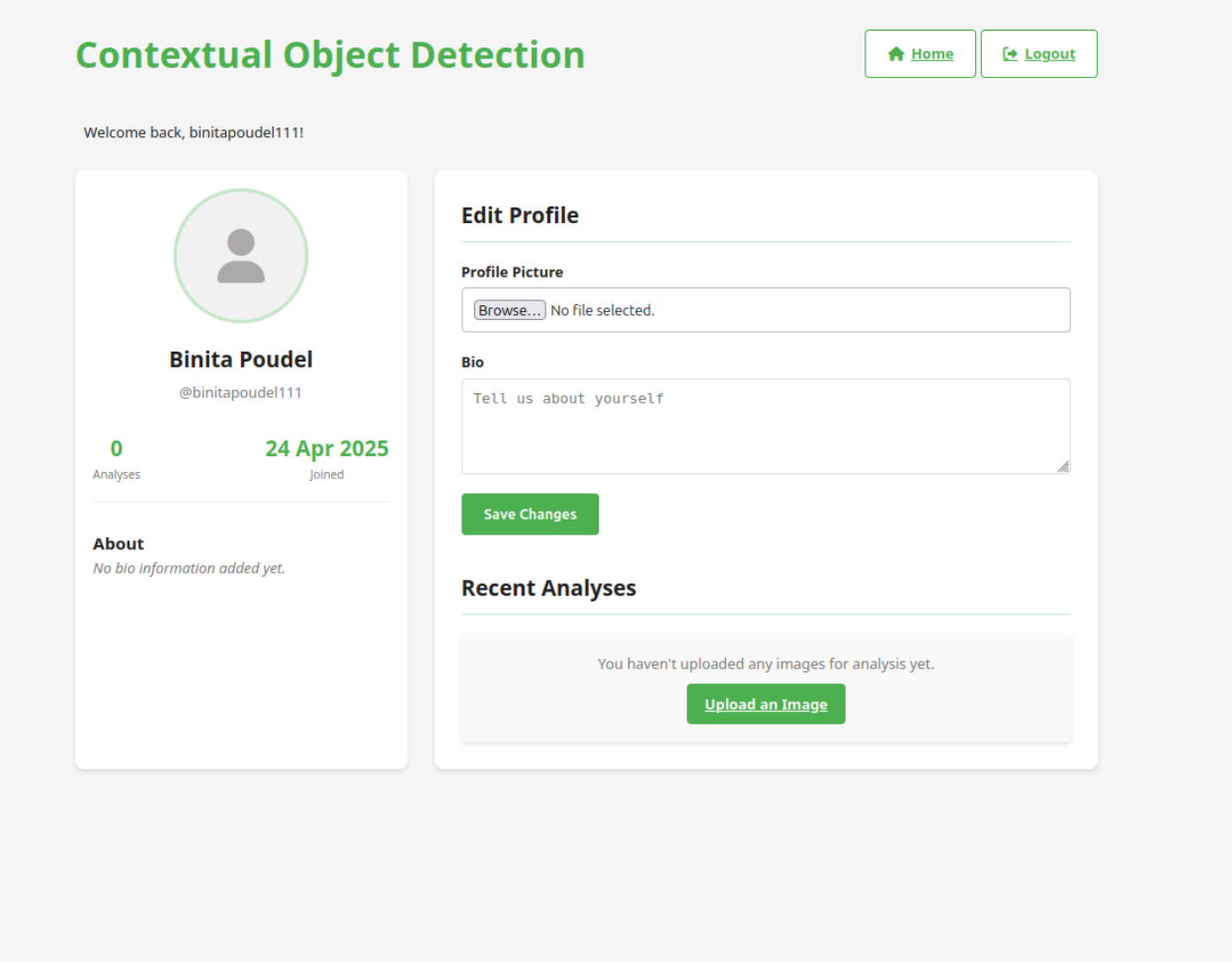


Figure 46: User access to their profile

4.2.7. Test Case 7: Check user can update their profile picture

Objectives	To check whether the user can update their profile picture during the session.
Action	User choose the image locally for using it as profile picture.
Expected Result	User should be able to set and update their profile picture.
Actual Result	Successfully set and updated the profile picture.
Conclusion	Test passed.

Table 11: Test 7: Check user can update their profile picture

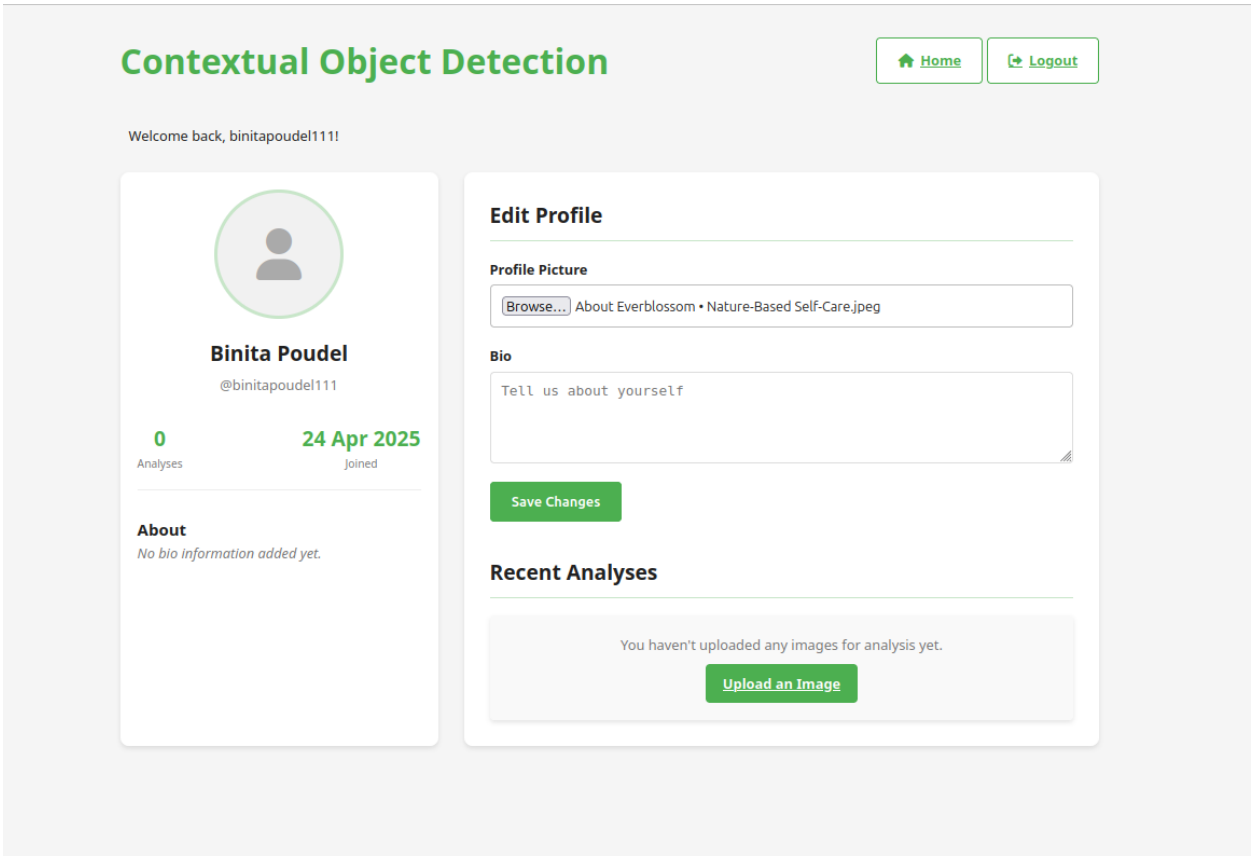


Figure 47: Choosing the profile picture

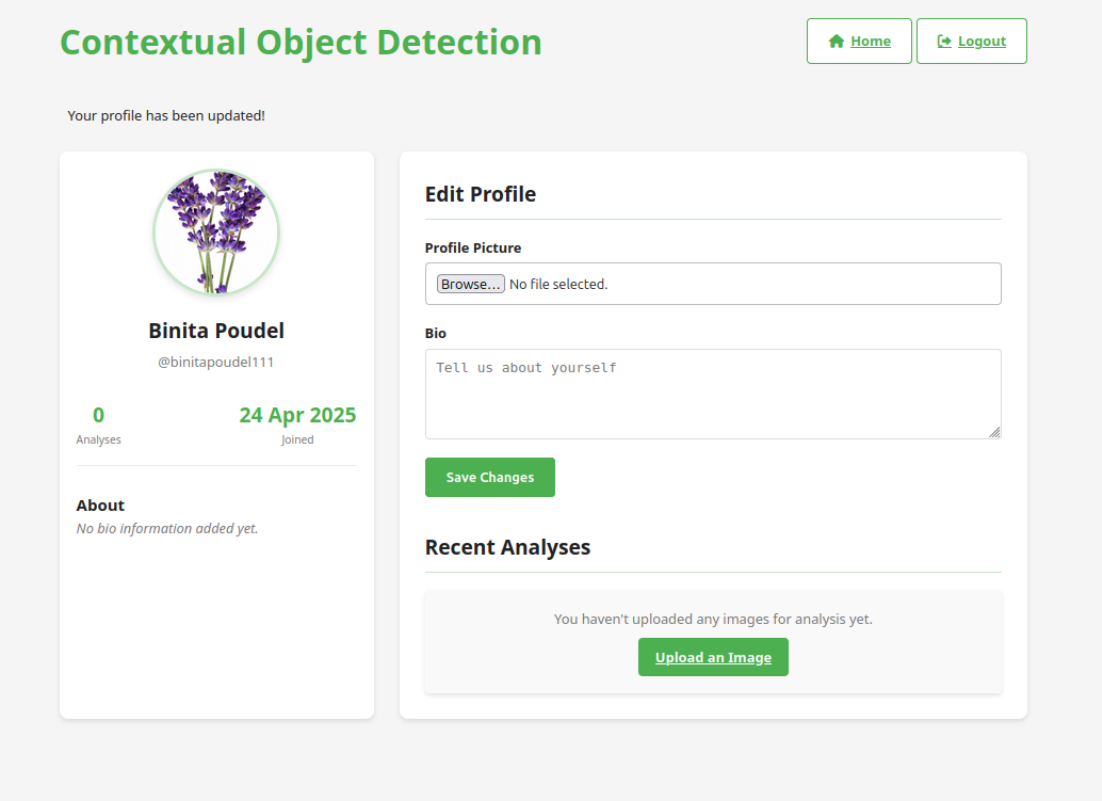


Figure 48: Profile picture updated

4.2.8. Test Case 8: Confirm whether the user can add bio to their profile.

Objectives	To check whether the user can add bio to their profile.
Action	User enters their bio and click the save changes button to set and update the bio
Expected Result	User should be able to set and update the user successfully.
Actual Result	User was able to set and update their bio
Conclusion	Test passed.

Table 12: Test 8: Confirm whether the user can add bio to their profile

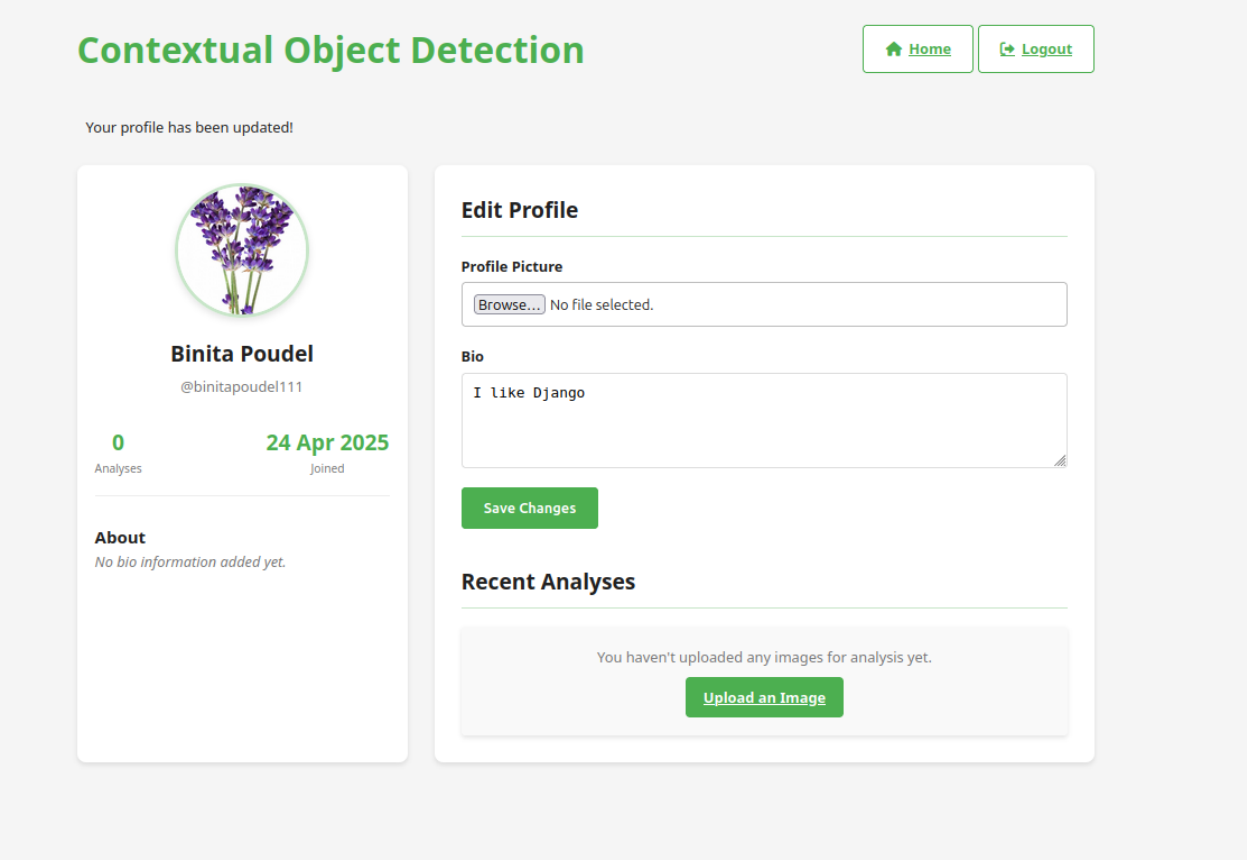


Figure 49: Adding bio to the profile

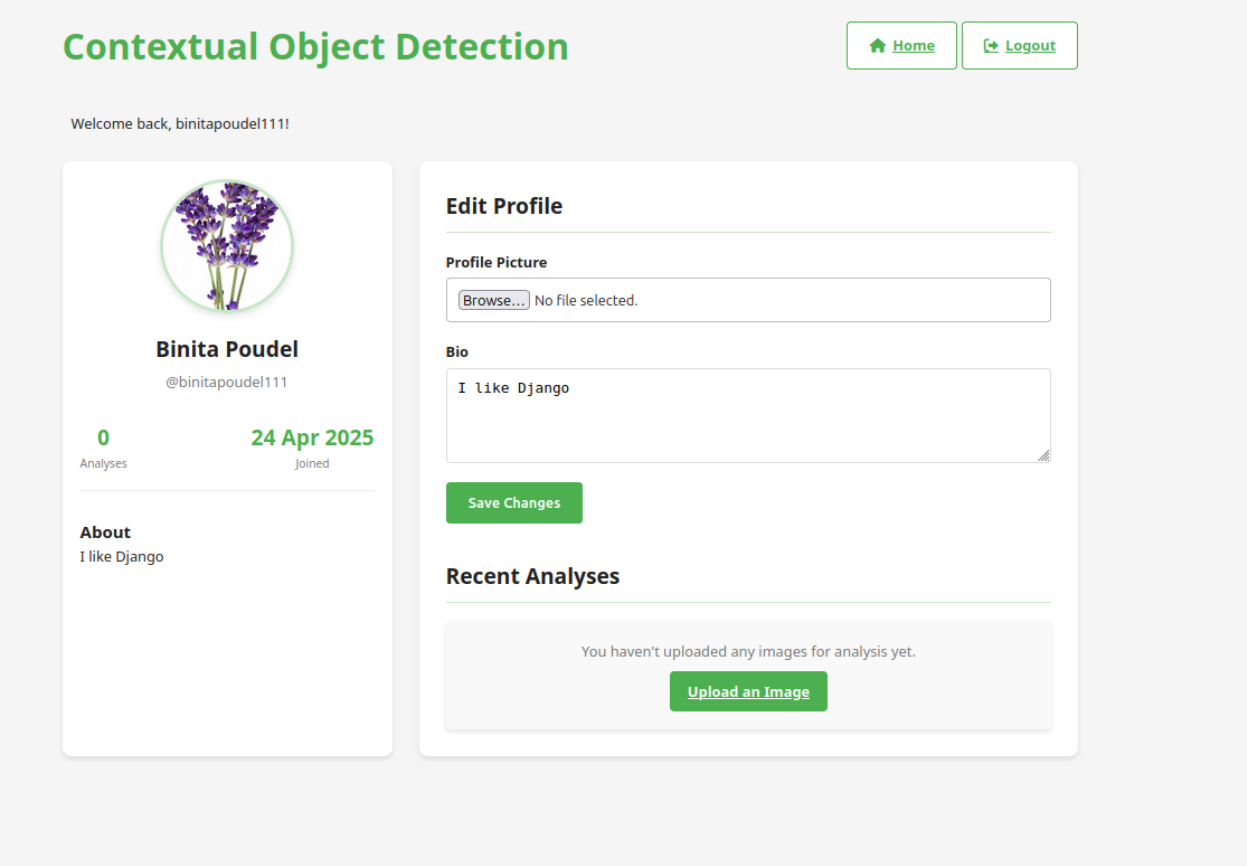


Figure 50: Bio updated successfully

4.2.9. Test Case 9: Check whether the chosen profile picture is updated in profile button of context generation page.

Objectives	To check whether the chosen profile picture is updated in profile button of context generation page.
Action	User choose the profile picture, saved the changes and return to main page.
Expected Result	The updated profile picture should be shown in context generation page.
Actual Result	Updated profile picture was shown in context generation page.
Conclusion	Test passed.

Table 13: Test 9: Check whether the chosen profile picture is update in profile button in context generation page

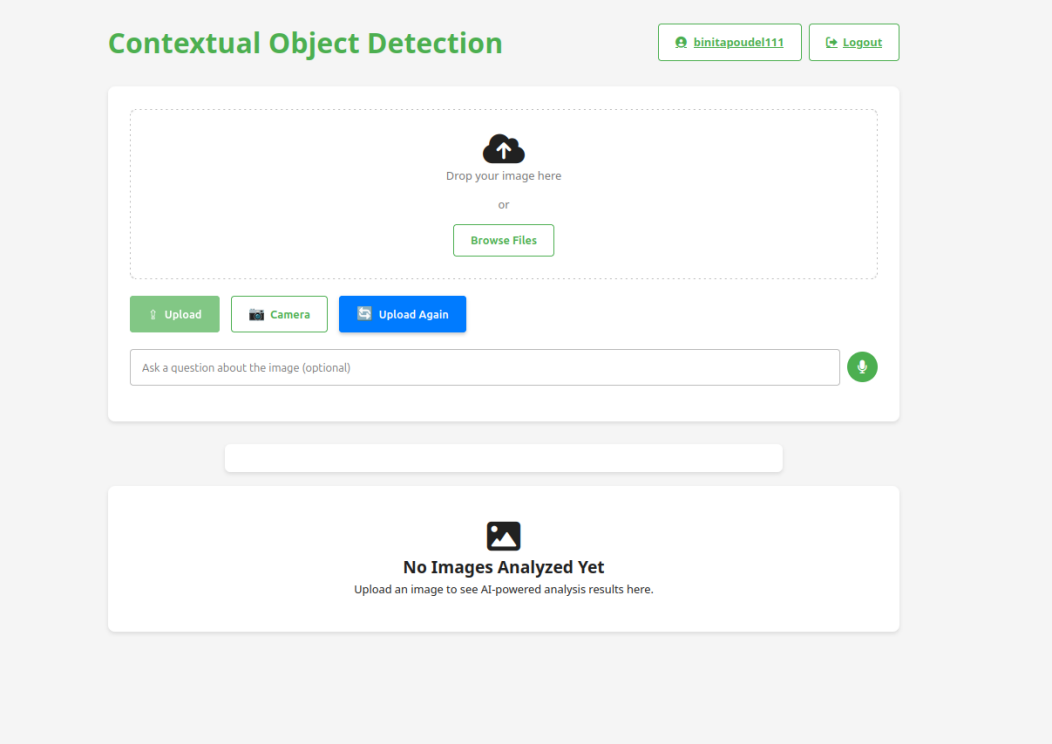


Figure 51: Main page before updating the profile picutre

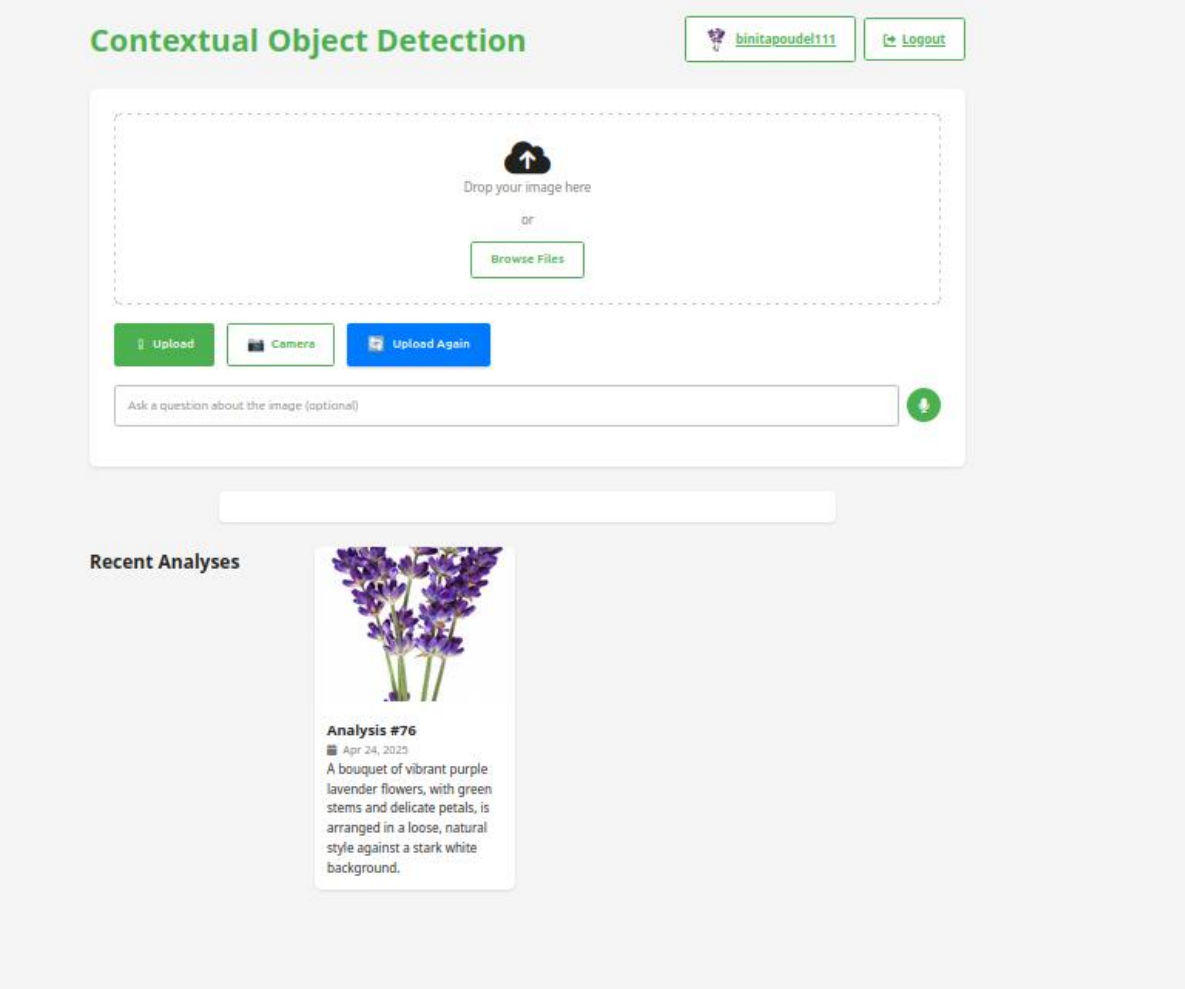


Figure 52: Context generation page after updating the proifle picture

4.2.10. Test Case 10: Check browsing the image functionality works.

Objectives	To check whether the user can choose the image from their device or not.
Action	Click on browse files button, choose the desired image.
Expected Result	User should be able to choose the desired image from their device.
Actual Result	User was able to choose the image locally using browse files button.
Conclusion	Test passed.

Table 14: Test 10: To check browsing the image functionality works

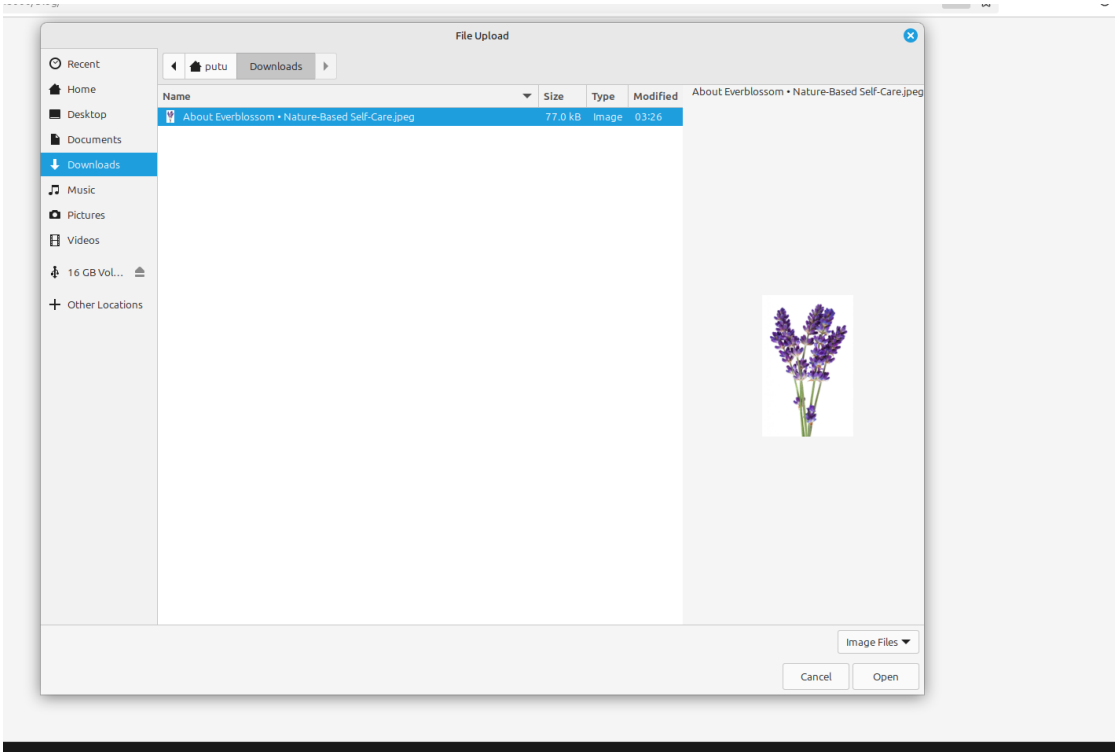


Figure 53: Choosing image for context generation

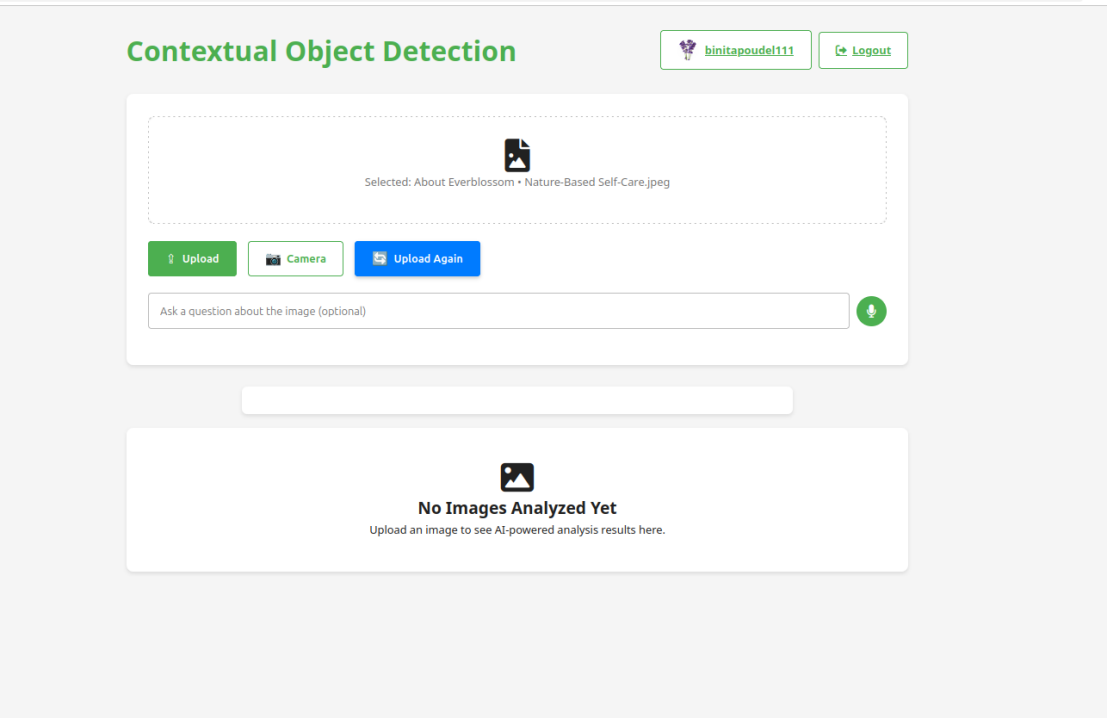


Figure 54: Image chosen successfully

4.2.11. Test Case 11: Confirm that the upload buttons works and context generation starts.

Objectives	To confirm whether the upload button works and context generation starts.
Action	User choose the image and then click on upload button with or without visual query to generate the context for the image.
Expected Result	The image should be successfully uploaded and context is generated within a short time.
Actual Result	Upload button worked and the image uploaded.
Conclusion	Test passed.

Table 15: Test 11: Confirm that the upload button works and context generation starts

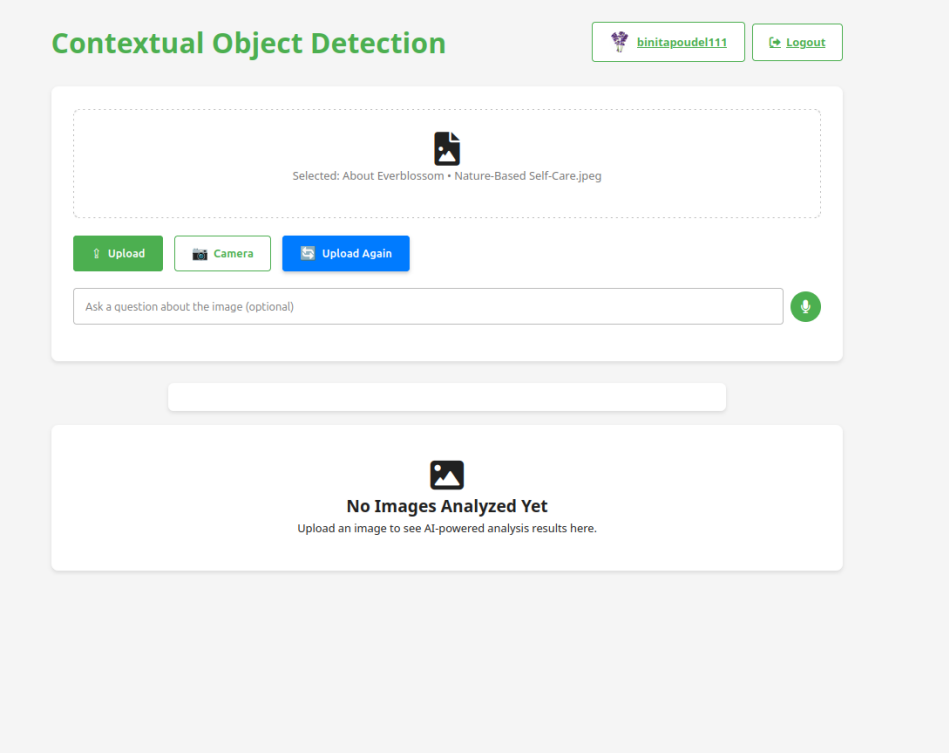


Figure 55: Uploading image

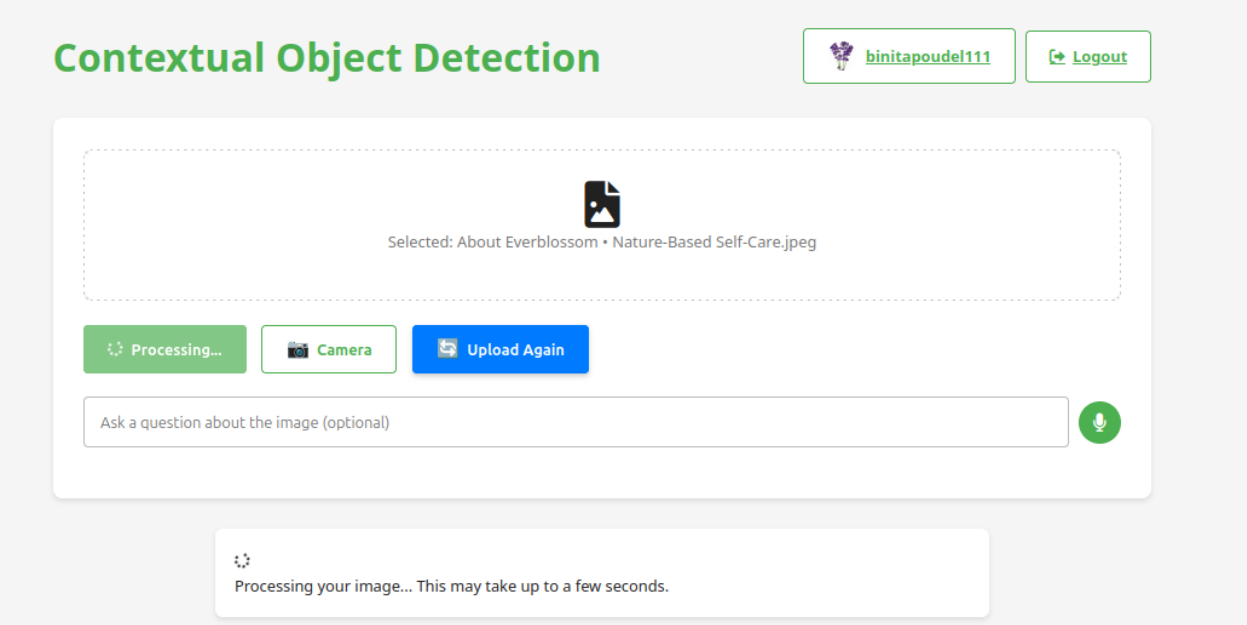


Figure 56: Image uploading

4.2.12. Test Case 12: Capture the image in real time and process it for context generation without visual query

Objectives	To capture the image in real time and process it for context without any visual query.
Action	User should capture the image using the camera button and provide it to the system for processing.
Expected Result	The user should be able to capture the image in real time successfully and process it using the model.
Actual Result	The user successfully captured the real time image and provided it to the system
Conclusion	Test passed.

Table 16: Test 12: Capture the image in real time and proces it for context geneation without visual query

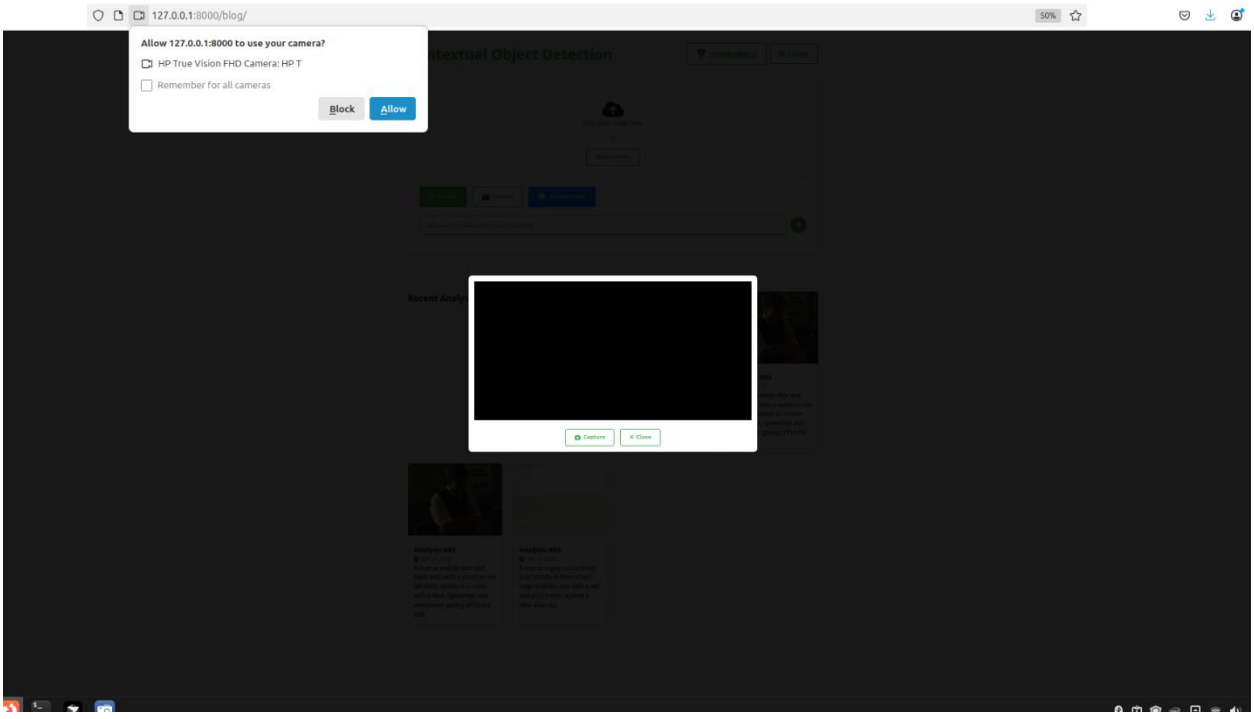


Figure 57: System asking for permission for accessing user's camera

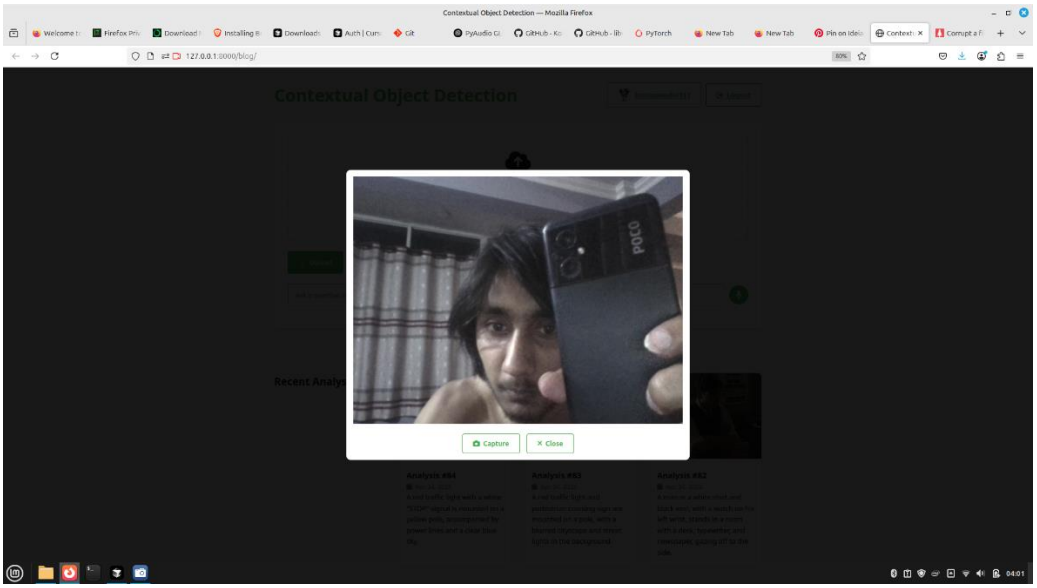


Figure 58: Capturing image in real time

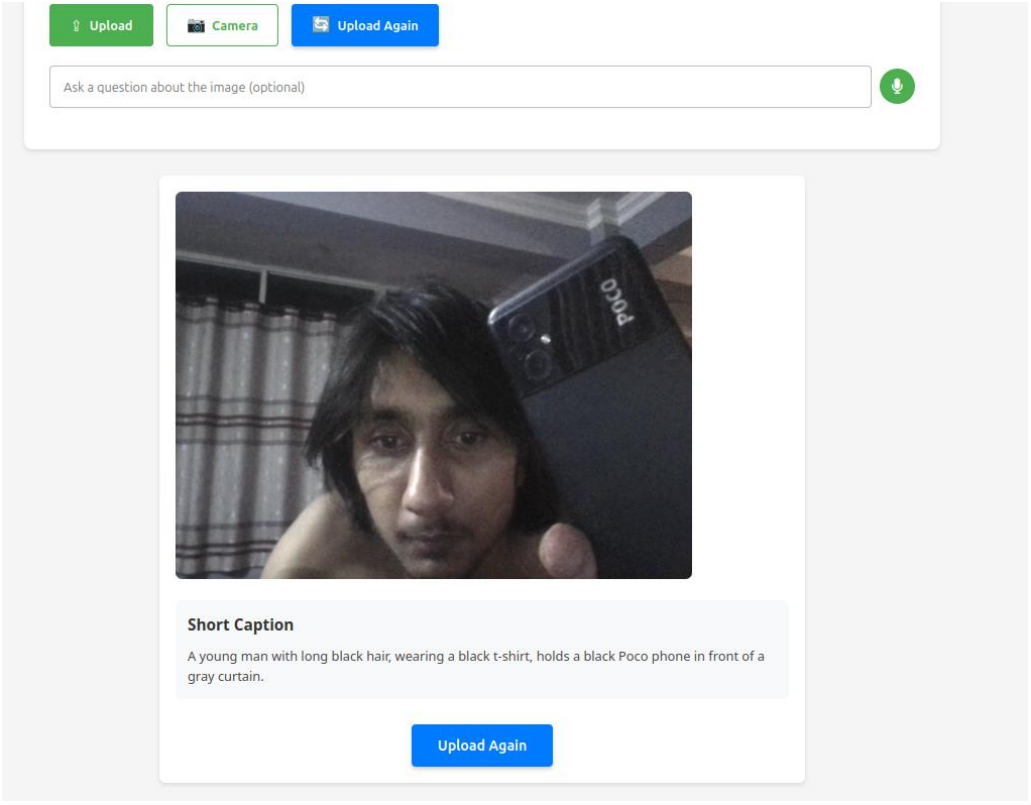


Figure 59: Generated context for captured image

4.2.13. Test Case 13: Capture the image in real time and process it for context generation with visual query

Objectives	To capture the image in real time and process it for context generation with visual query.
Action	User should provide the visual query first and then capture the image using the camera functionality.
Expected Result	The user should be able generate the context with visual query for real time captured image.
Actual Result	Successfully provided query for the real time captured image.
Conclusion	Test passed.

Table 17:Test 13: Capture the image in real time and process it for context with visual query

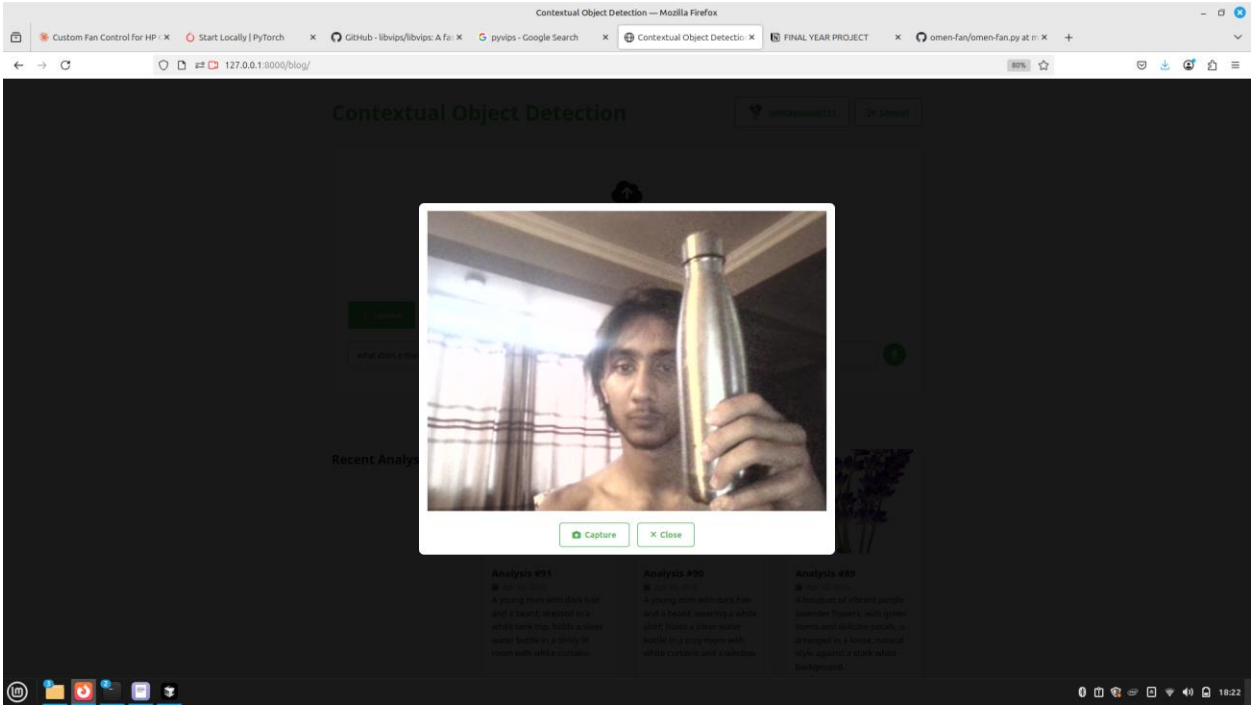


Figure 60: Capturing image in real time with subject

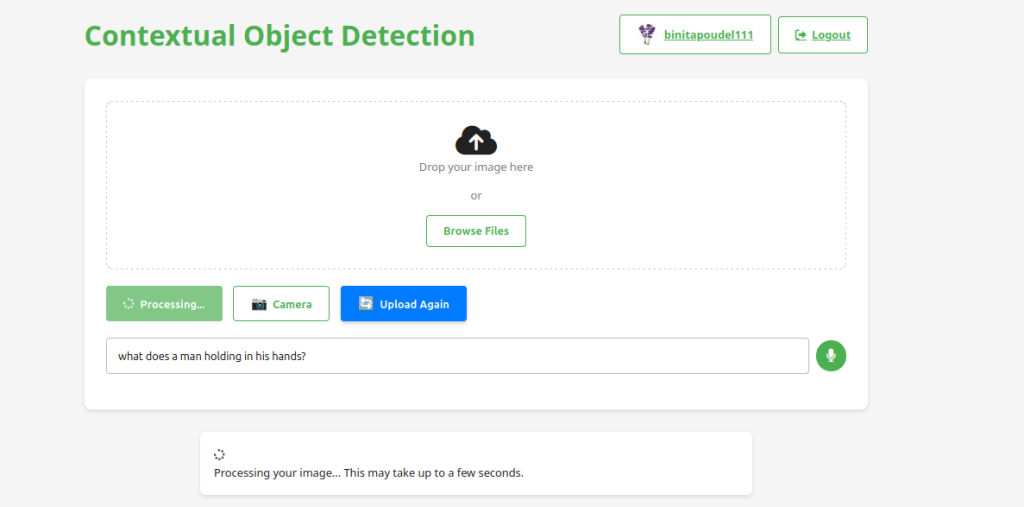


Figure 61: Providing captured image with visual query

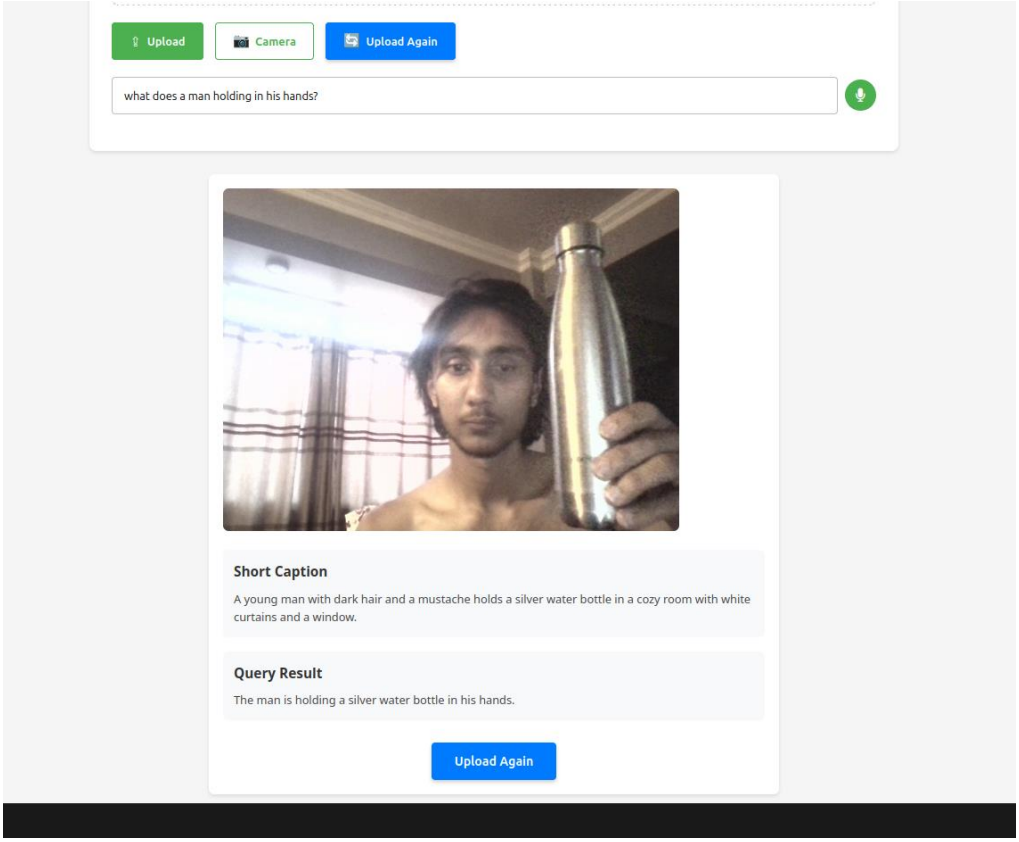


Figure 62: Result for real time capture image with visual query

4.2.14. Test Case 14: Consecutive image upload for context generation without visual query

Objectives	To confirm the system ability to process consecutive images without visual query provided.
Action	User should consecutively provide around 3 images for context generation without visual query.
Expected Result	System should generate the context for consecutive images within short time frame efficiently.
Actual Result	Successfully generated the context for the consecutive images uploaded to the system by the user.
Conclusion	Test passed.

Table 18: Test 14: Consecutive image upload for context generation without visual query

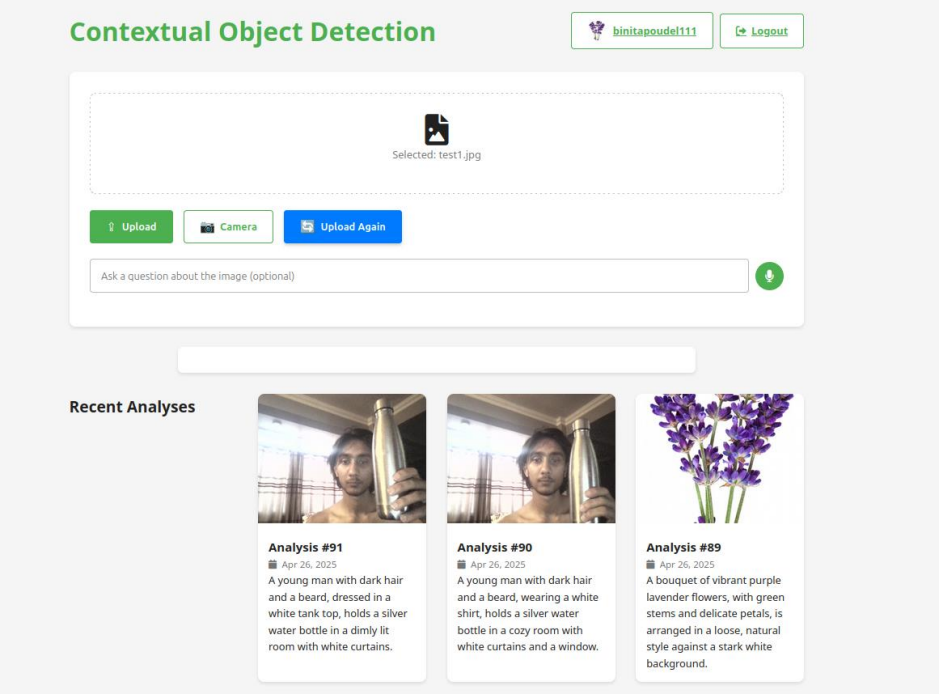


Figure 63: Uploading first image

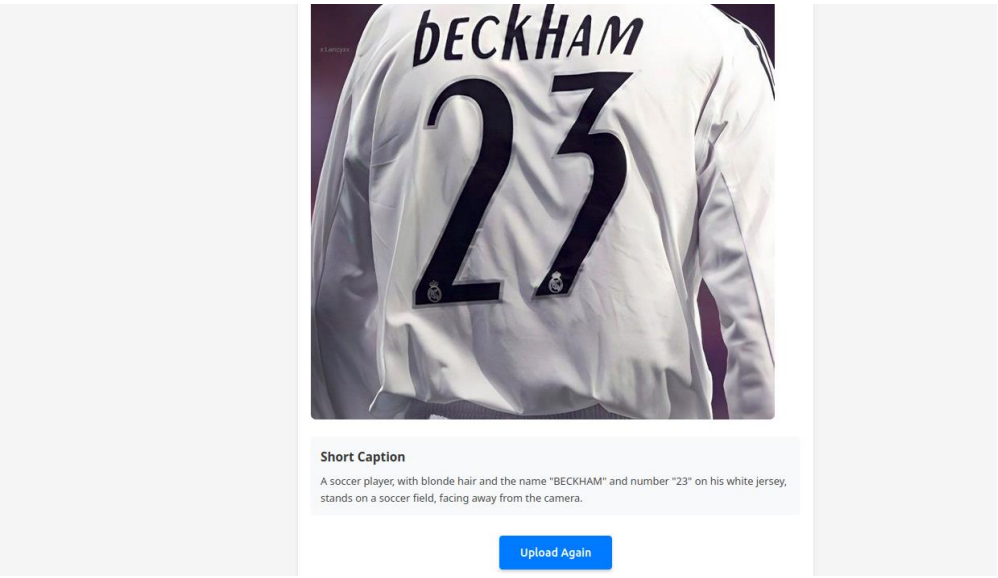


Figure 64: Result for the first image

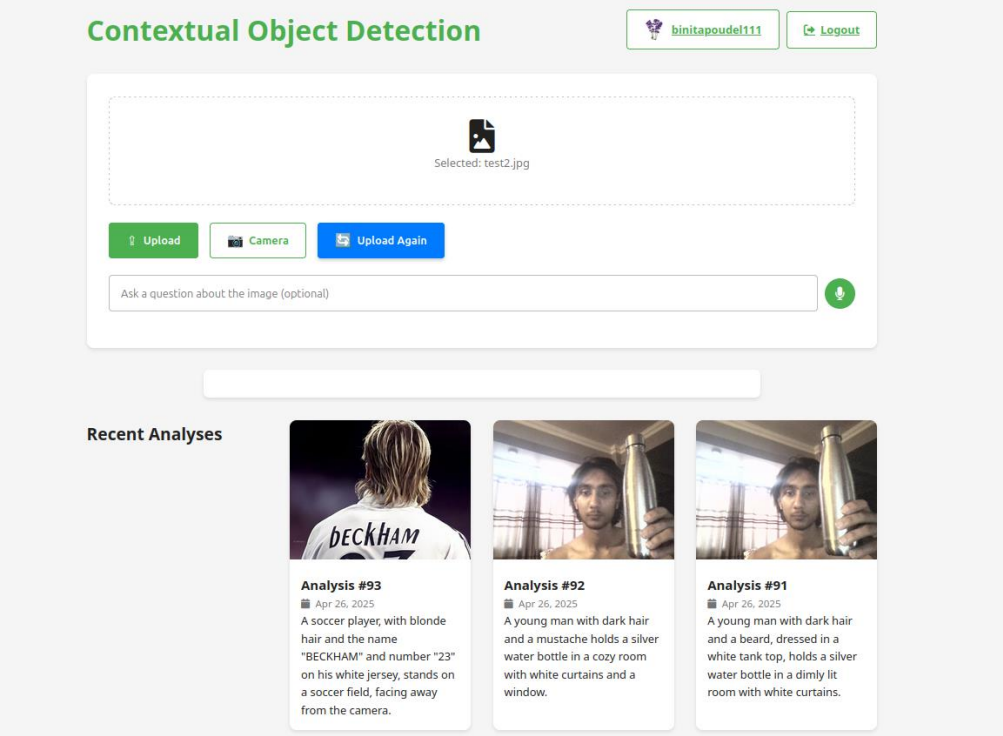


Figure 65: Uploading second image

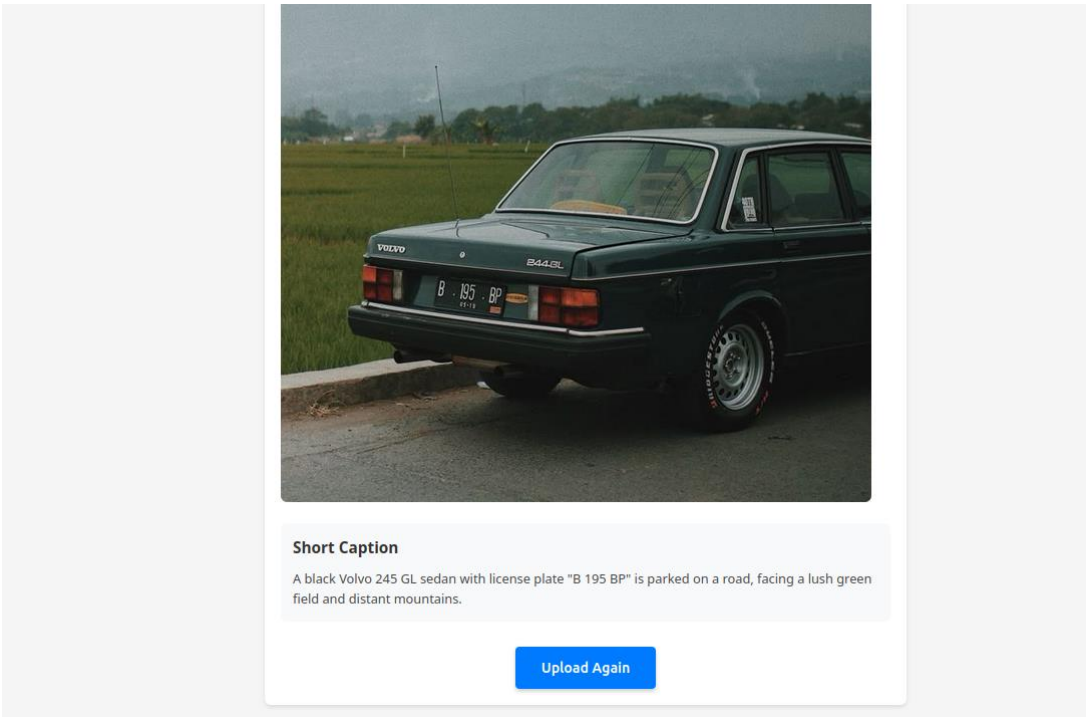


Figure 66: Context for second image

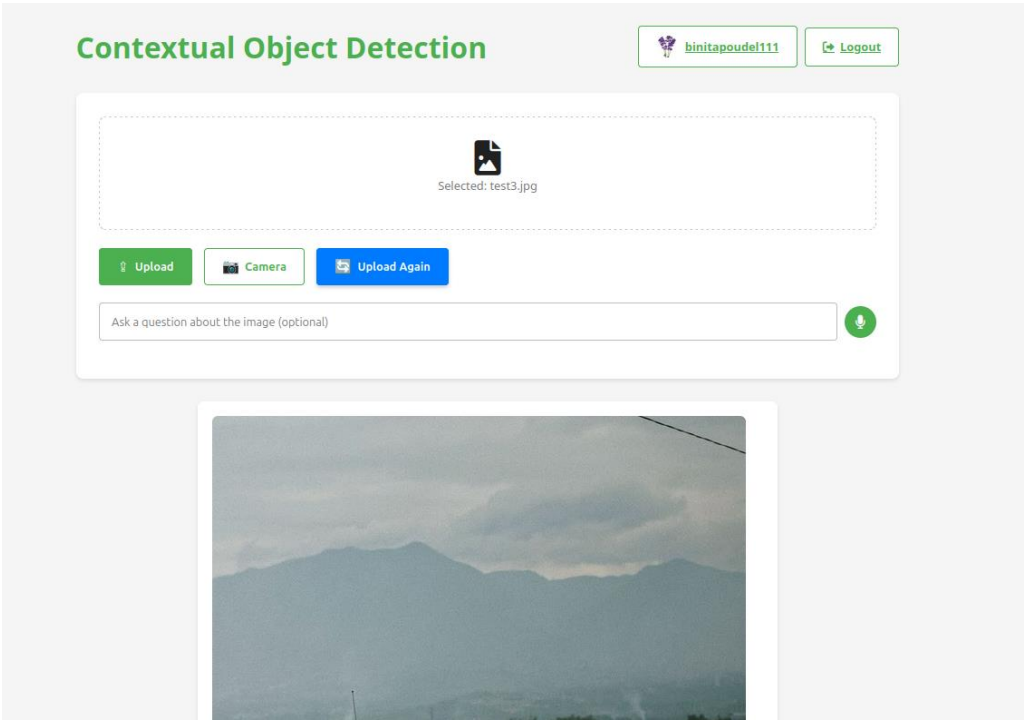


Figure 67: Uploading third consecutive image

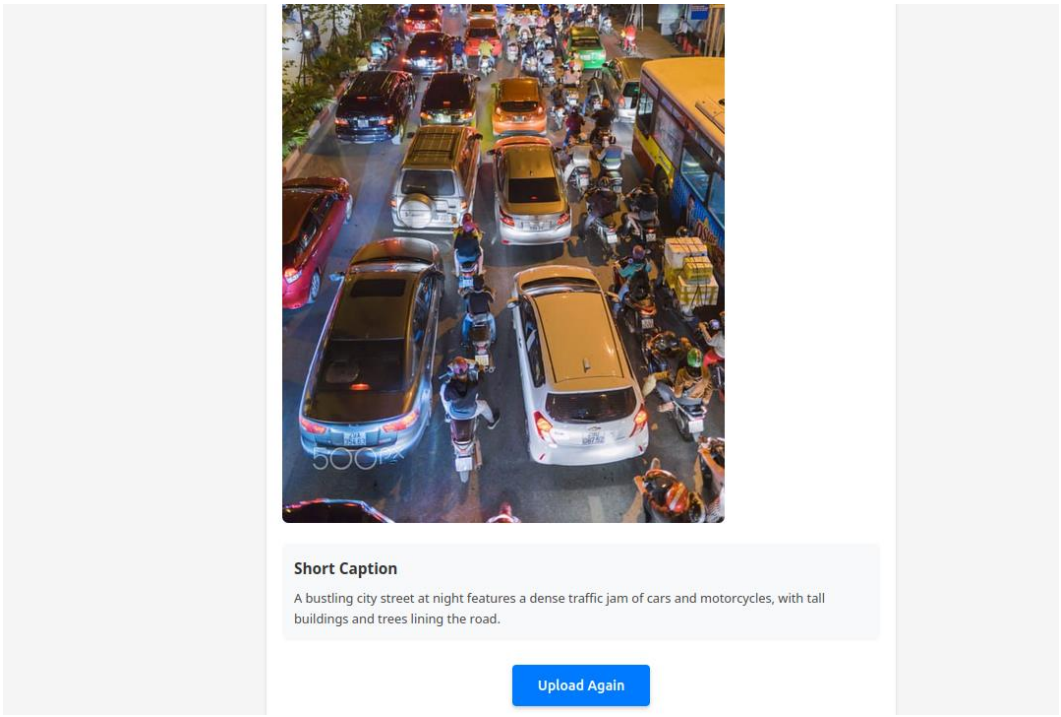


Figure 68: Context for third image

4.2.15. Test Case 15: Consecutive image upload and capture for context generation with visual query.

Objectives	To consecutively upload the images and capture for context generation with visual query.
Action	User provided the images by uploading and capturing in real time (3 images) for context generation with visual query about the image.
Expected Result	The system should be able to handle the consecutive upload along with visual query to generate the context.
Actual Result	The system was able to handle consecutive images with visual queries efficiently.
Conclusion	Test passed.

Table 19: Test 15: Consecutive image upload and capture for context generation with visual qeuery

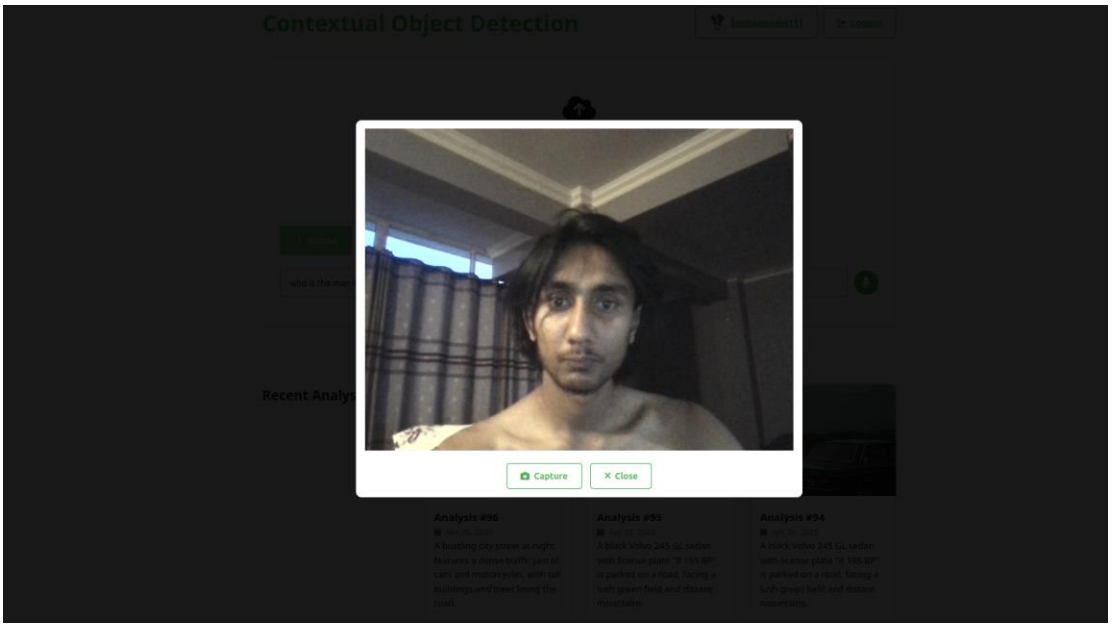


Figure 69: Capturing image for visual query

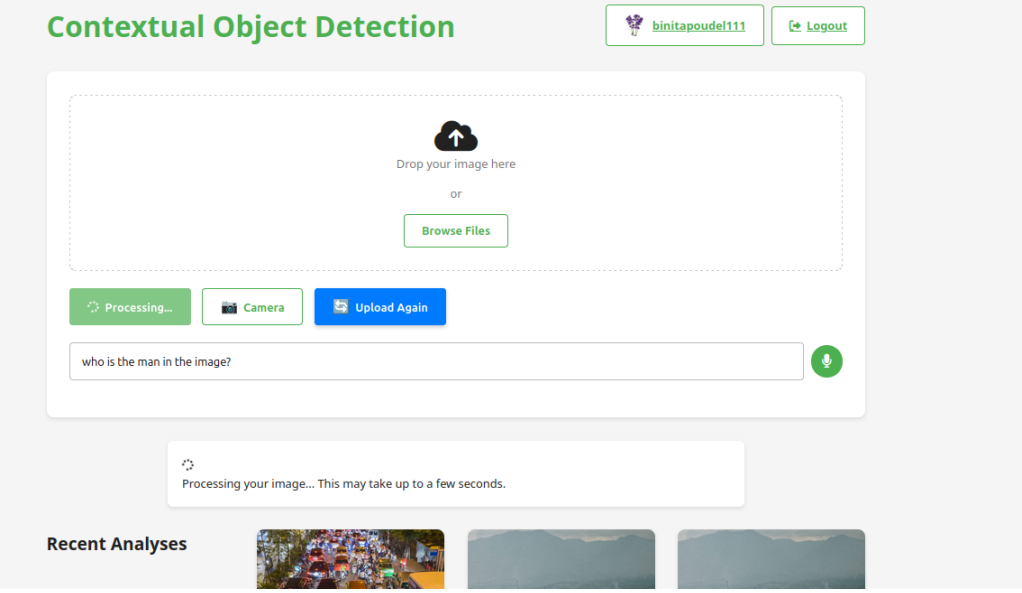


Figure 70: Processing first image

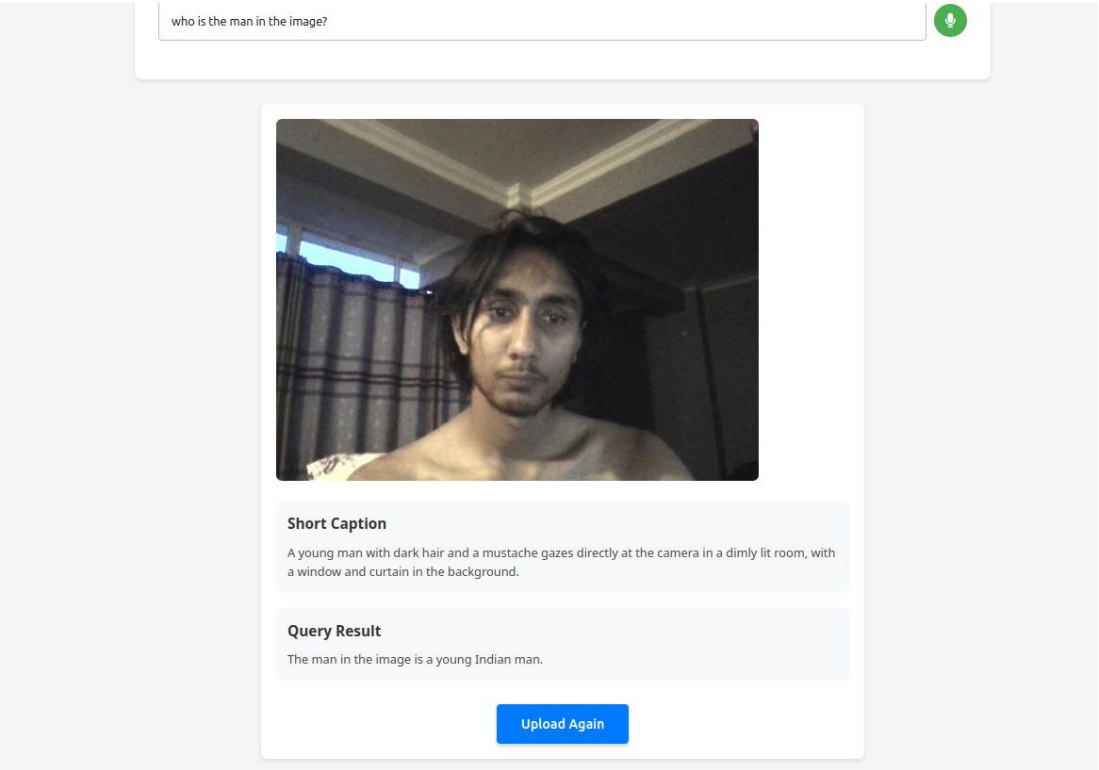


Figure 71: Context generated for first image

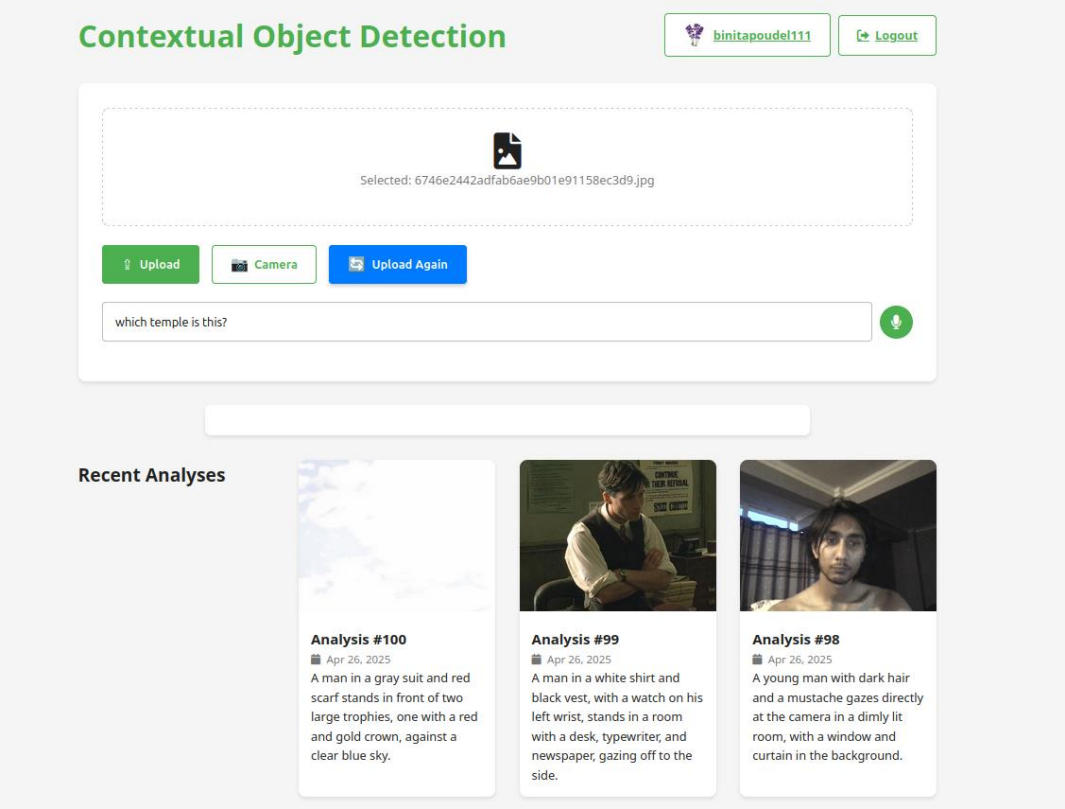


Figure 72: Uploading second image

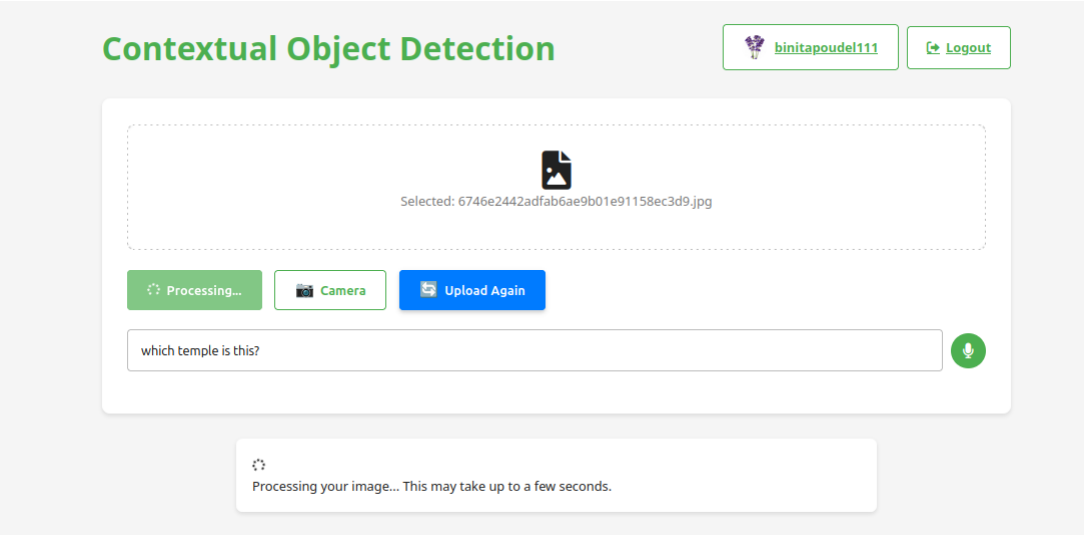


Figure 73: Processing second image

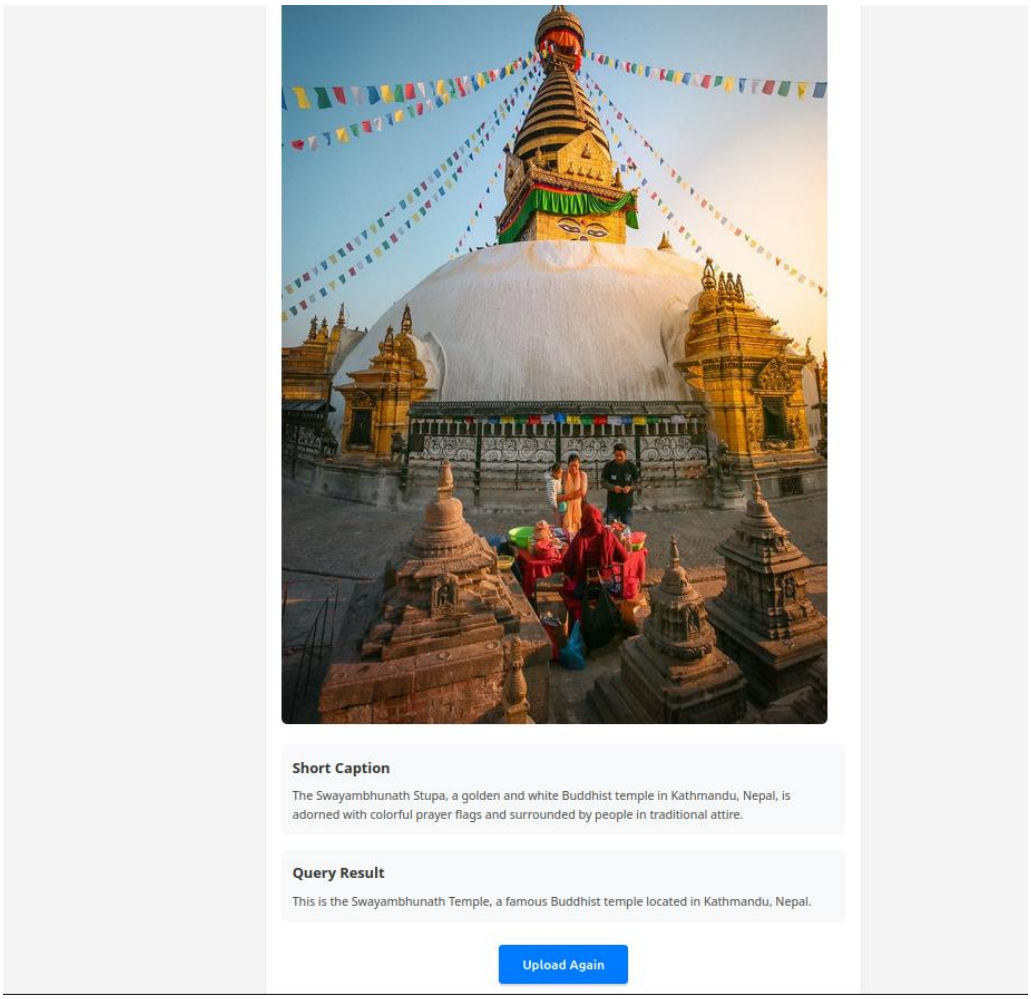


Figure 74: Context generated for second image

4.2.16. Test Case 16: Invalid file upload for context generation

Objectives	To upload the invalid file with extension .pdf for context generation to the system.
Action	User upload the pdf file to the context generation system.
Expected Result	The system should provide the error message showing that the file is not supported.
Actual Result	The system didn't allow the upload of the invalid image.
Conclusion	Test passed.

Table 20: Test 16: Invalid file upload for context generation

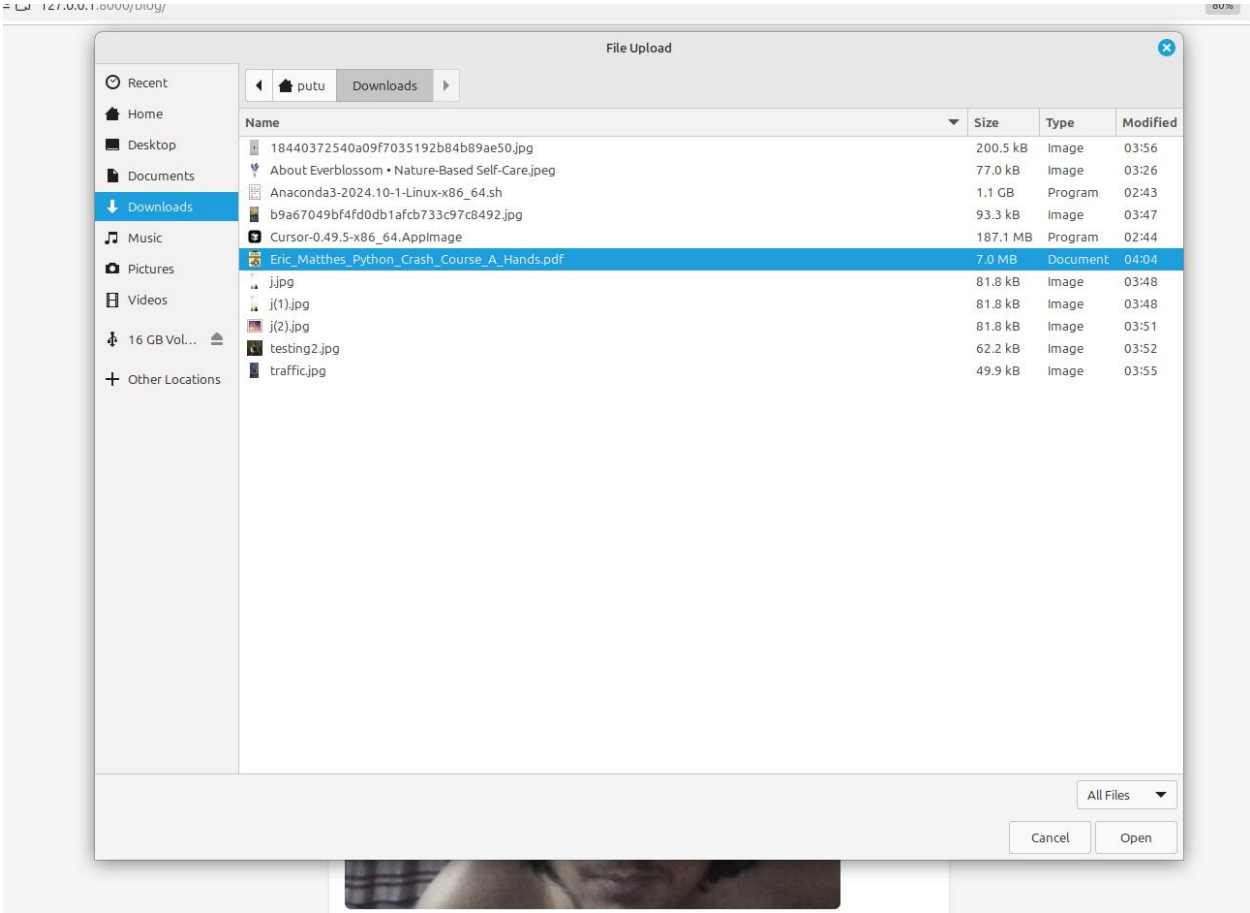


Figure 75: Invalid file upload

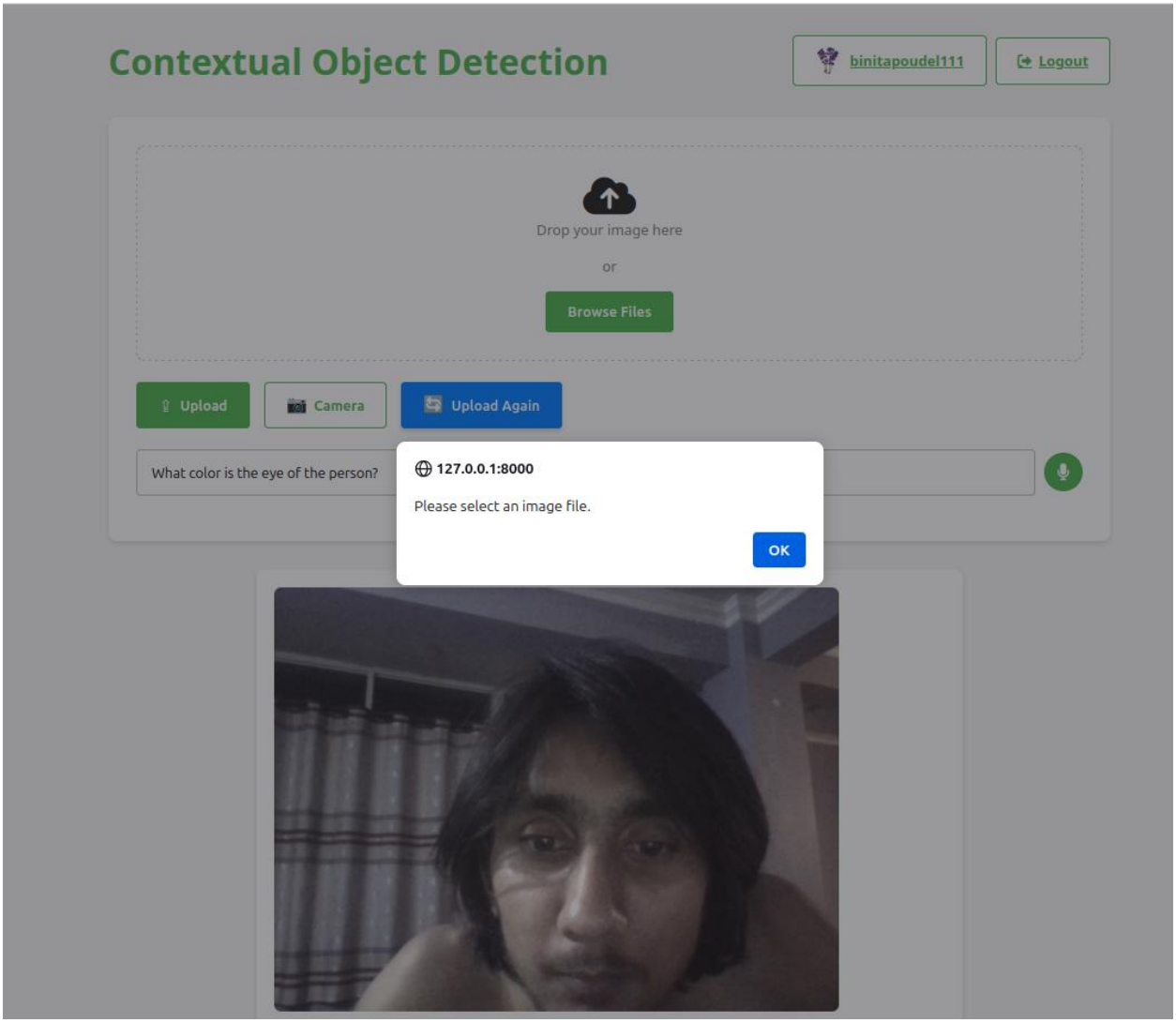


Figure 76: System doesn't allow choosing the invalid file

4.2.17. Test Case 17: Corrupted Image upload for context generation

Objectives	To upload the corrupted image for context generation
Action	User upload the corrupted image file for context generation
Expected Result	The system should not process the corrupted image.
Actual Result	The image didn't process the image and showed the processing failed message.
Conclusion	Test passed.

Table 21: Test 17: Corrupted image upload for context generation

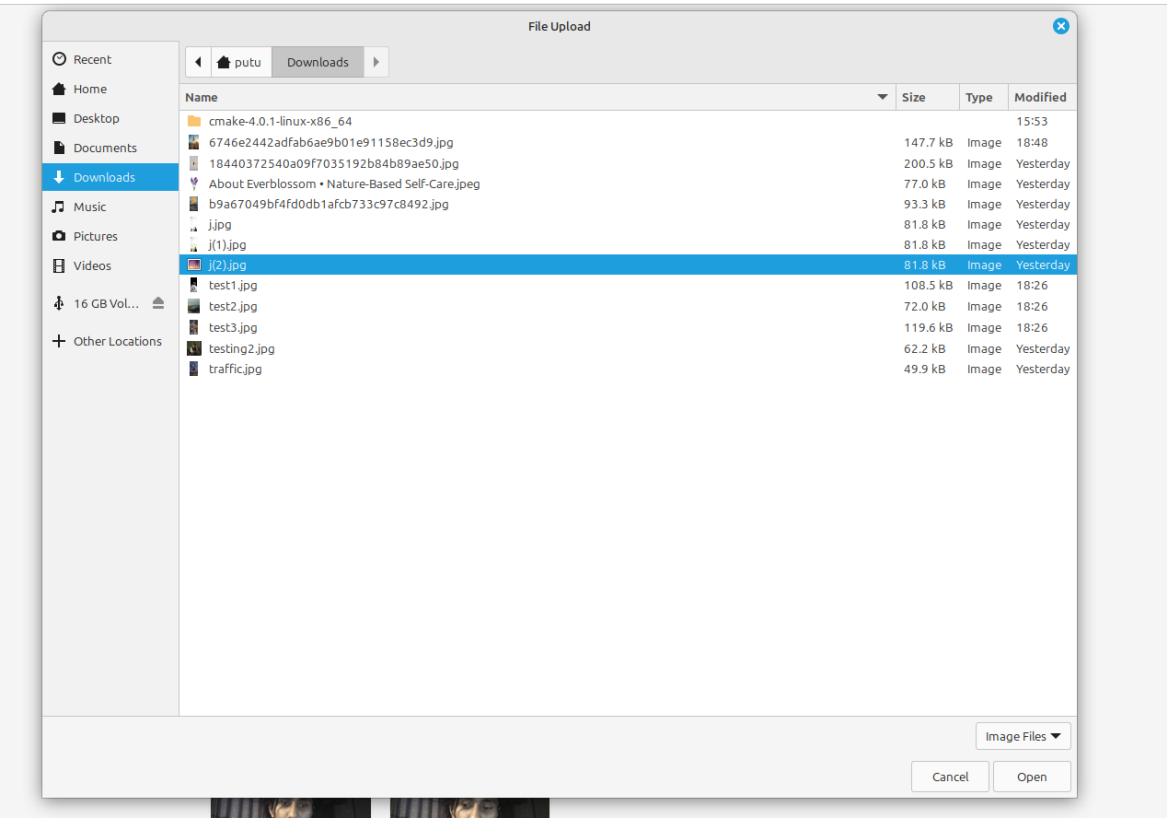


Figure 77: Choosing corrupted image

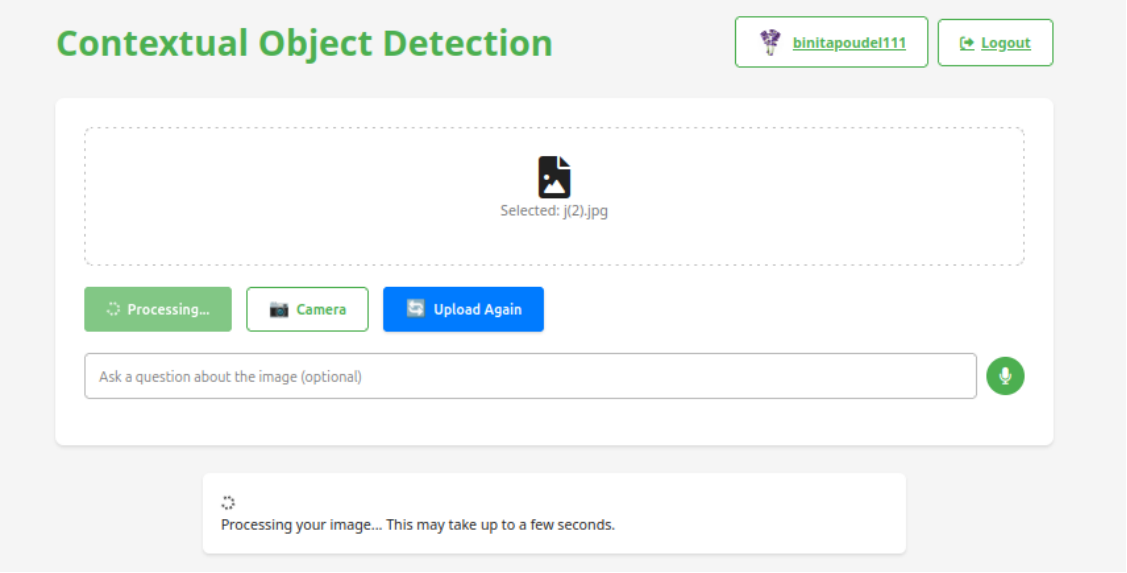


Figure 78: Uploading invalid image

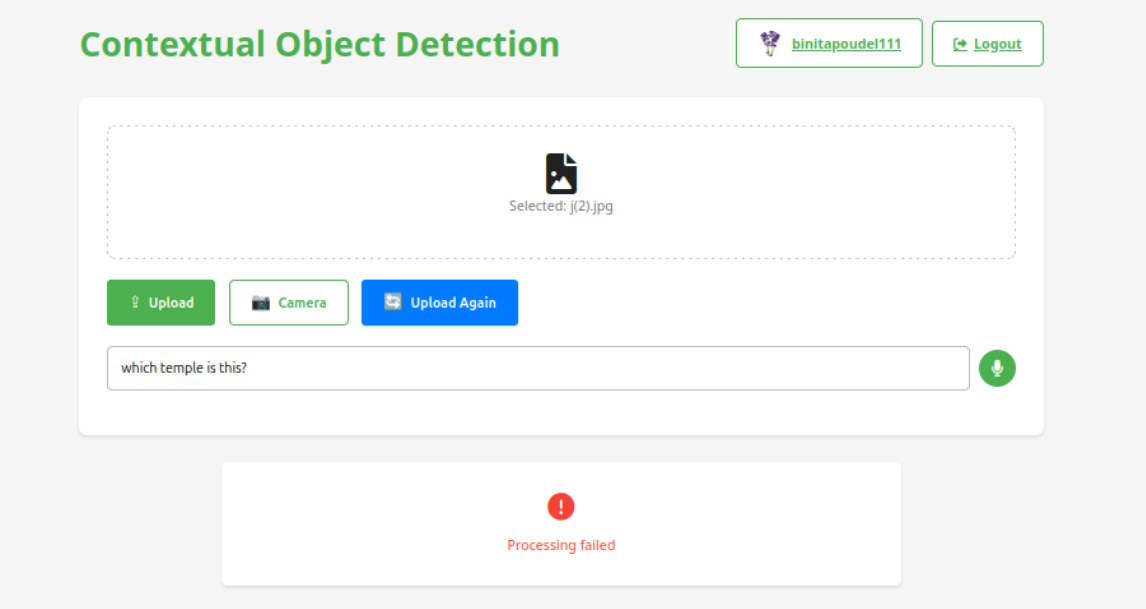


Figure 79:System response to corrupted image

```
INFO:blog.views:Processing image for user: binitapoudel111
[26/Apr/2025 13:14:08] "POST /blog/process-image/ HTTP/1.1" 200 126
[26/Apr/2025 13:14:10] "GET /blog/check-job/83d901a8-1787-4866-ae5e-ce9ebb693a7e/ HTTP/1.1" 200 69
[26/Apr/2025 13:14:12] "GET /blog/check-job/83d901a8-1787-4866-ae5e-ce9ebb693a7e/ HTTP/1.1" 200 69
[26/Apr/2025 13:14:14] "GET /blog/check-job/83d901a8-1787-4866-ae5e-ce9ebb693a7e/ HTTP/1.1" 200 50
```

Figure 80: System response without error to the corrupted image

4.2.18. Test Case 18: Check speech-to-text functionality for user query

Objectives	To check the speech-to-text functionality for user query.
Action	User should use the microphone button in the system to provide the visual query through speech to text.
Expected Result	System should effectively convert the speech to the text and use it as the query for generating the context for the image.
Actual Result	System successfully converted the speech to text and used it as query for the image uploaded.
Conclusion	Test passed.

Table 22: Test 18: Check speech-to-text functionality for user query

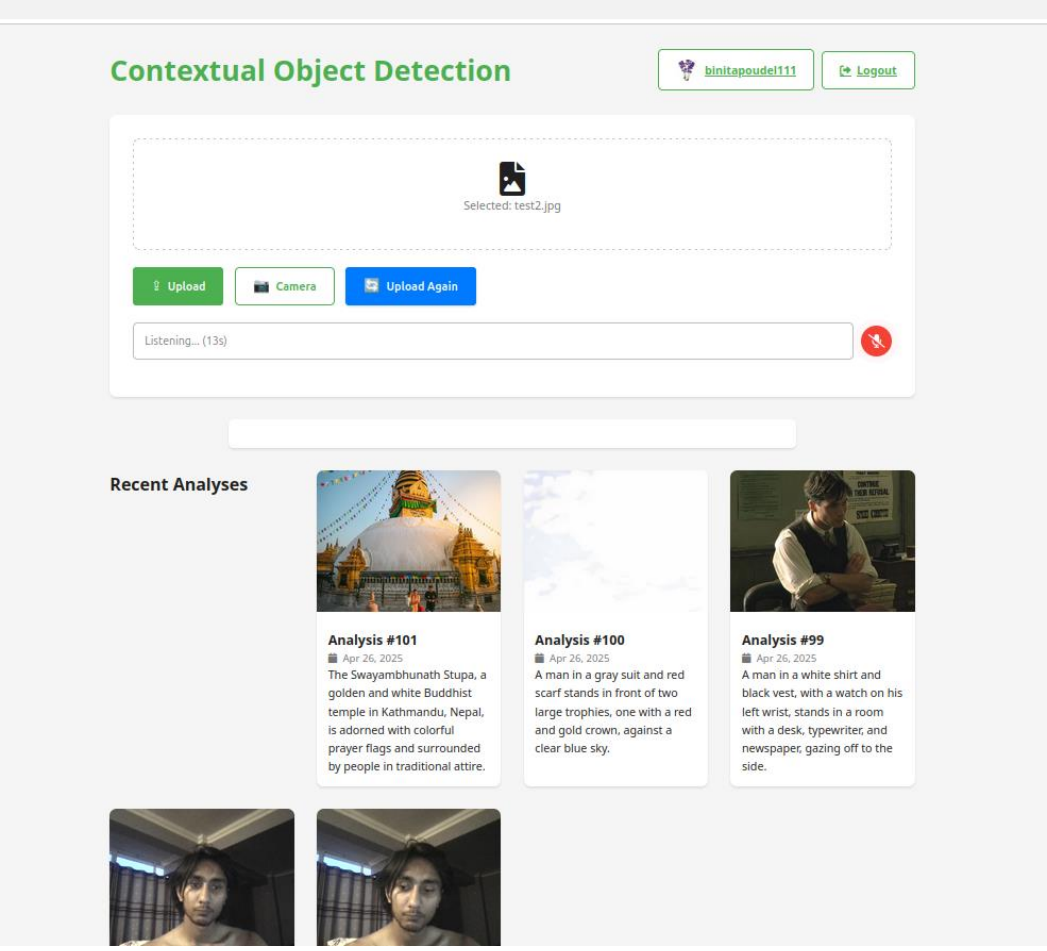


Figure 81: Using microphone for providing visual query through stt

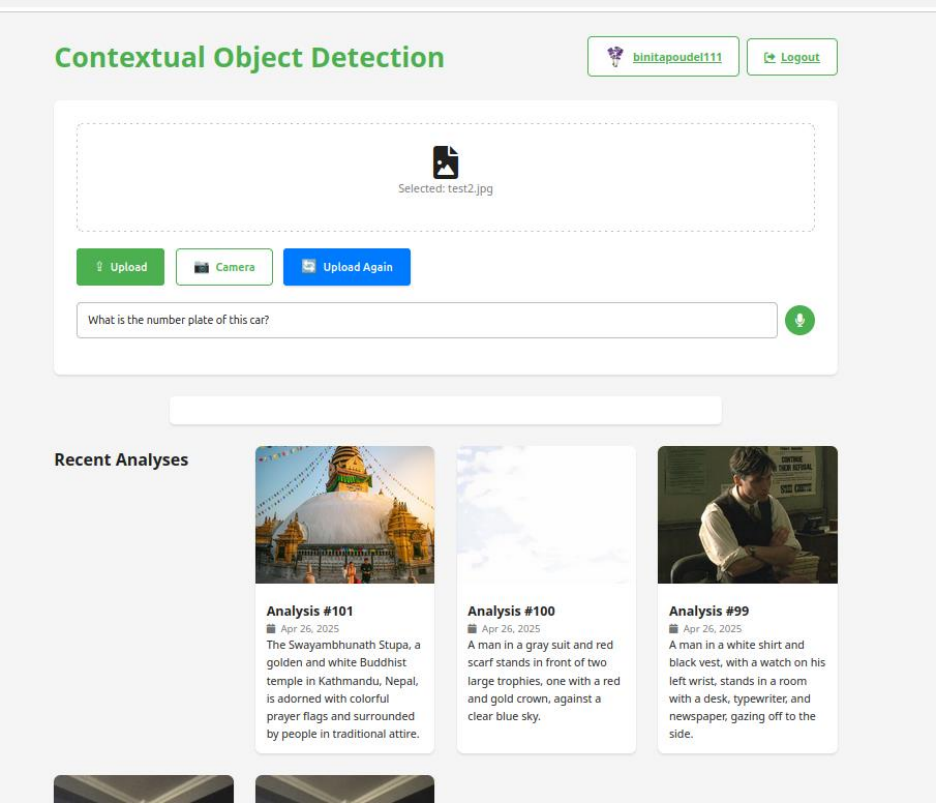


Figure 82: User's speech converted to text correctly

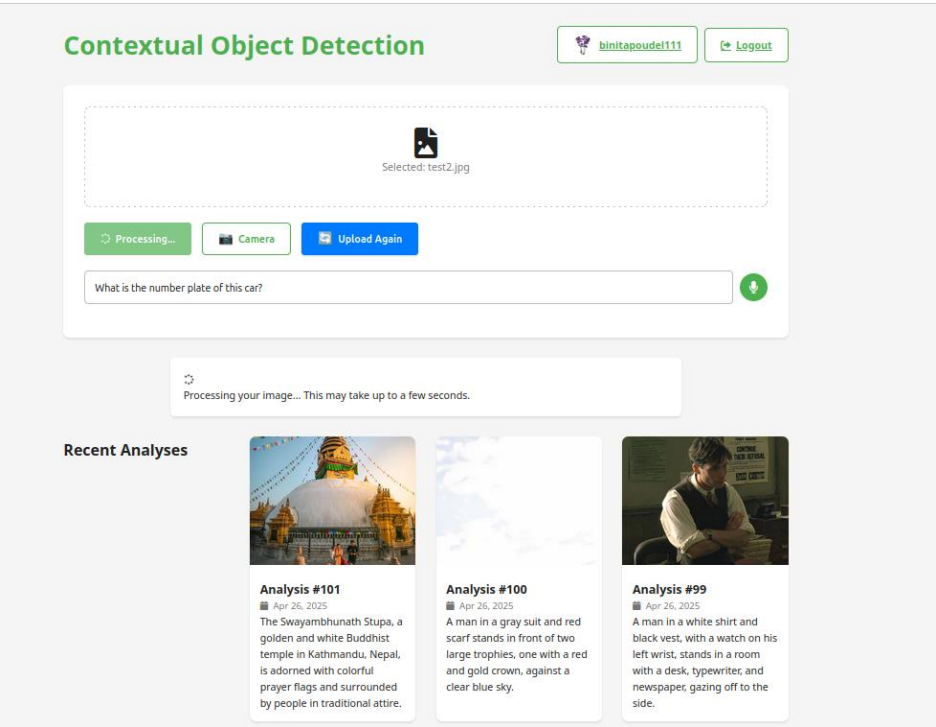


Figure 83: Processing the image with stt visual query

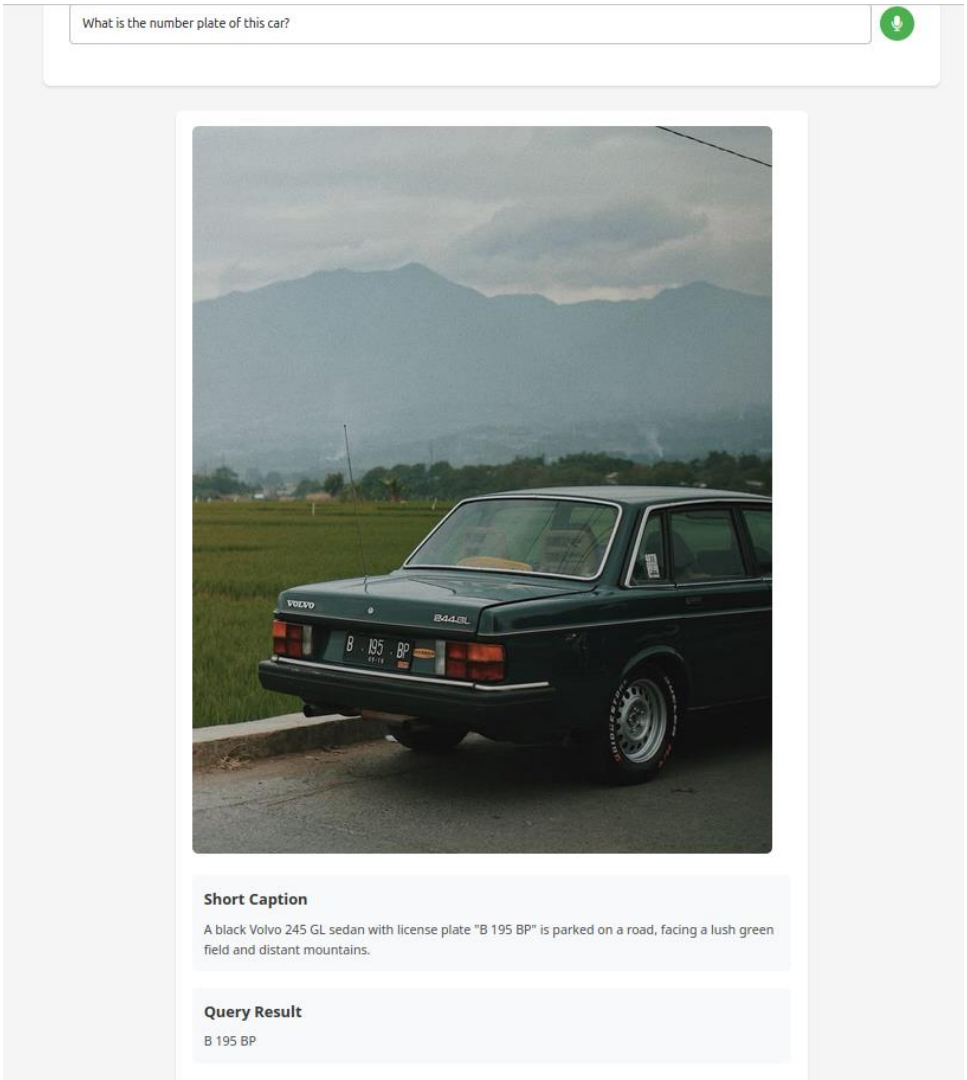


Figure 84: Context generated with answer to user's visual query

4.2.19. Test Case 19: Visual query for image without speech-to-text functionality

Objectives	To provide the visual query for the image without speech-to-text functionality.
Action	User should type the query for the image after uploading the image.
Expected Result	The system should use the query provided by the user to provide the answer to the visual query.
Actual Result	System effectively took the typed visual query and used it for context generation.
Conclusion	Test passed.

Table 23: Test 19: Visual query for image without speech-to-text functionality

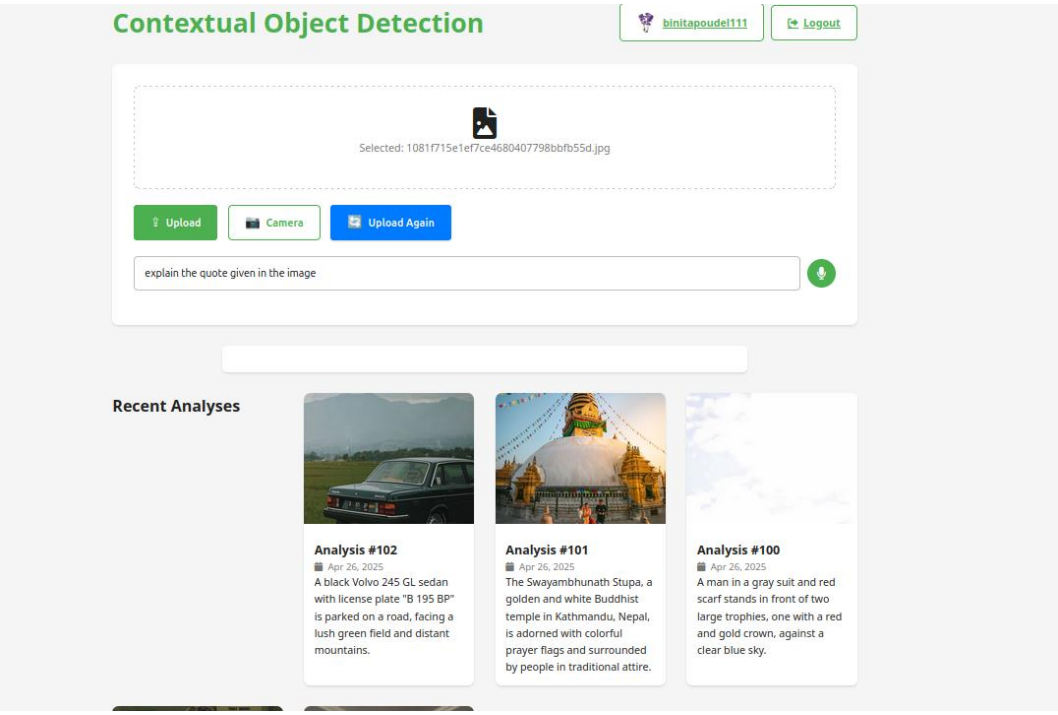


Figure 85: Uploading image with written visual query

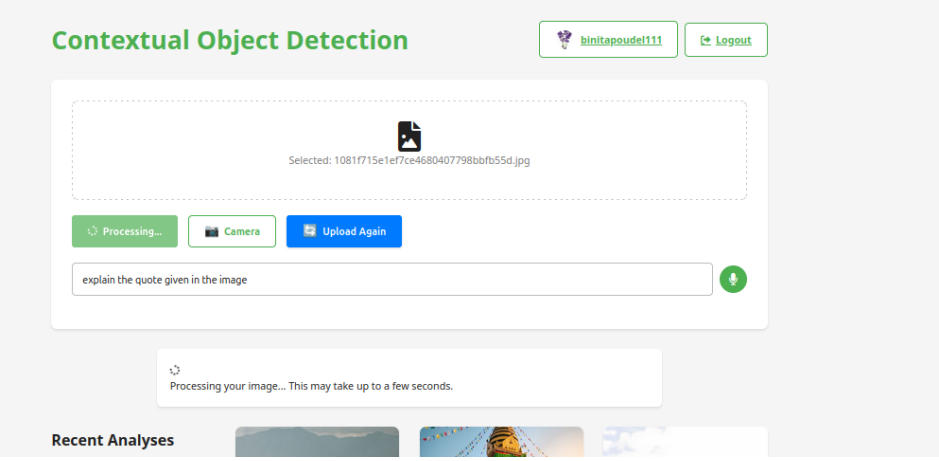


Figure 86: Processing the image

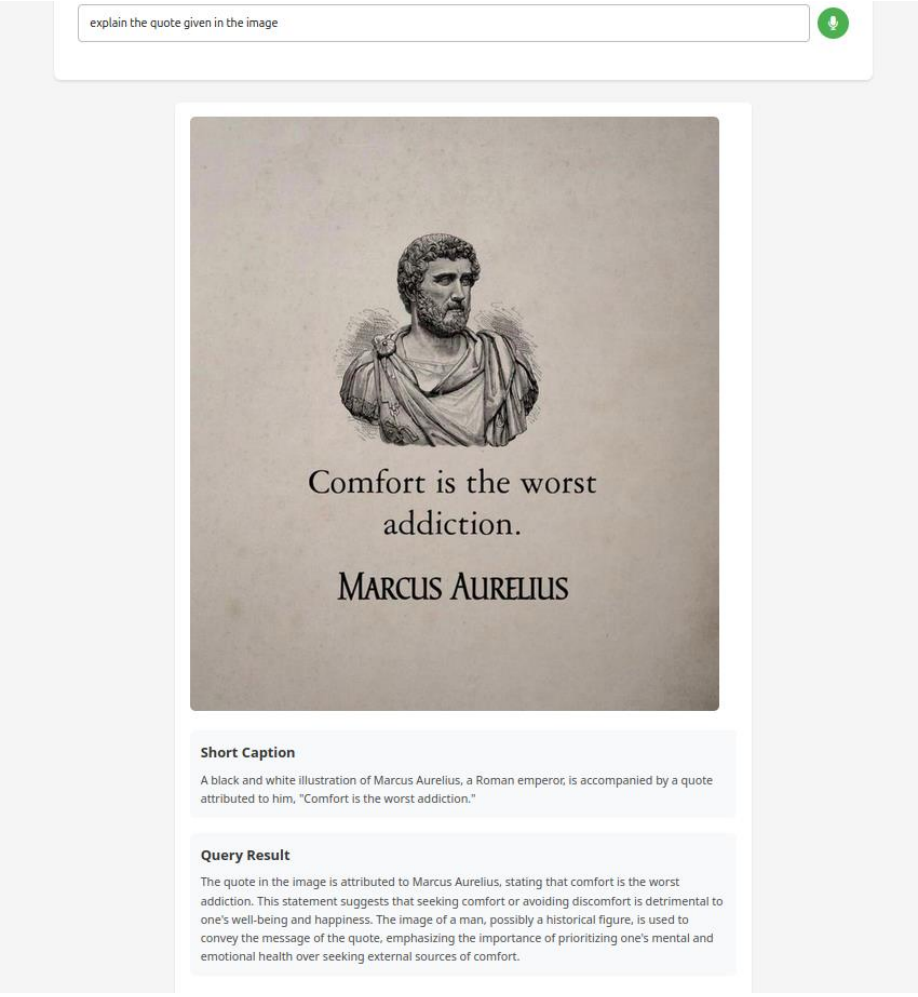


Figure 87: Generated caption with visual query

4.2.20. Test Case 20: Short context generation for the image and time taken

Objectives	To generate only short caption for the image and check the time taken for generation.
Action	User should upload the image and provide to the system without any visual query.
Expected Result	The system should be able to generate the short context for the image within few seconds.
Actual Result	The system generated the short context for the image within 6 seconds.
Conclusion	Test passed.

Table 24: Test 20: Short context generation for the image and time taken

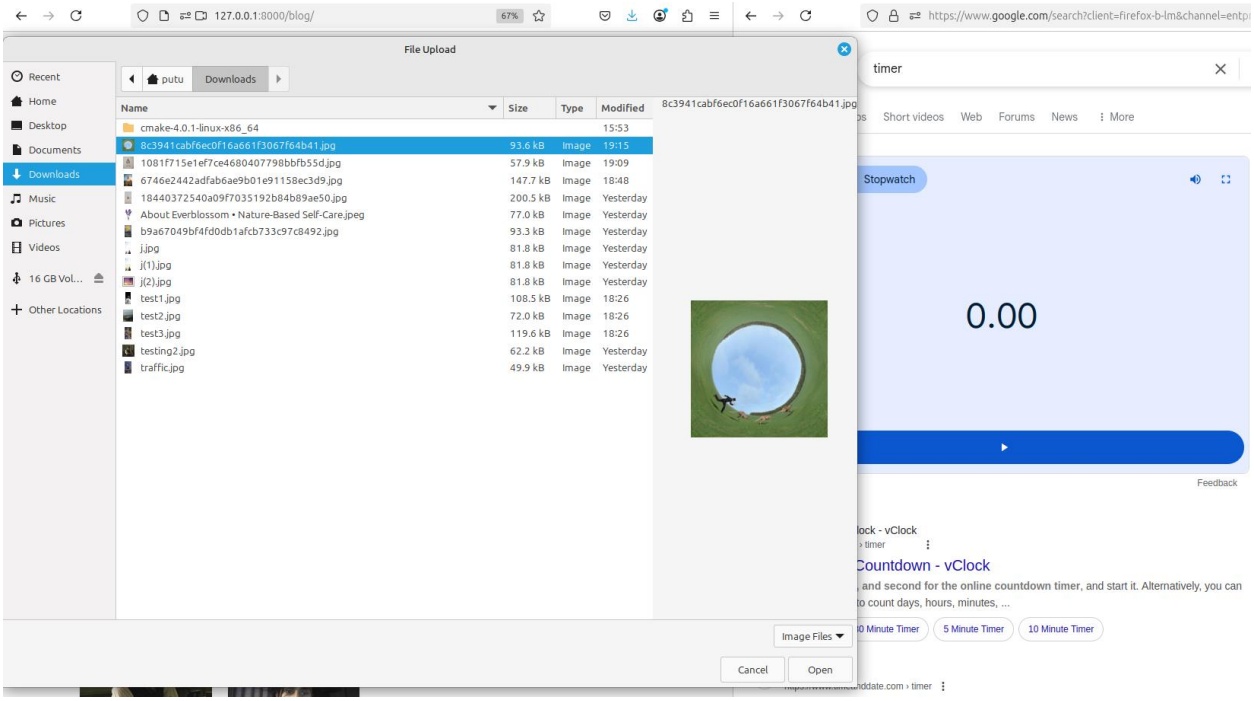


Figure 88:Uploading the image for short context generation

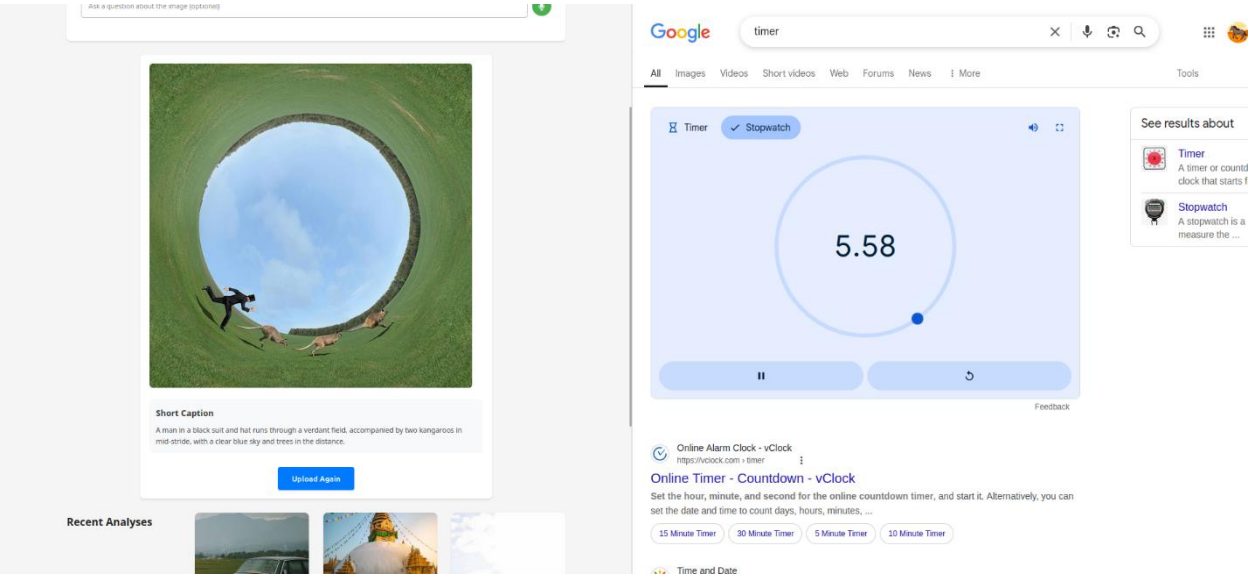


Figure 89: Generated short context and time taken

4.2.21. Test Case 21: Short context and visual query answering for the image and time taken

Objectives	To check the generation of the short context and visual query answering to the image and measuring the time taken.
Action	User upload the image along with the visual query through typing or speech-to-text.
Expected Result	The system should generate the short caption, answer to the visual query within few seconds for the image.
Actual Result	The system generated the context for the image with visual query's answer in 8 seconds.
Conclusion	Test passed.

Table 25: Test 21: Short context and visual query answering for the image and time taken

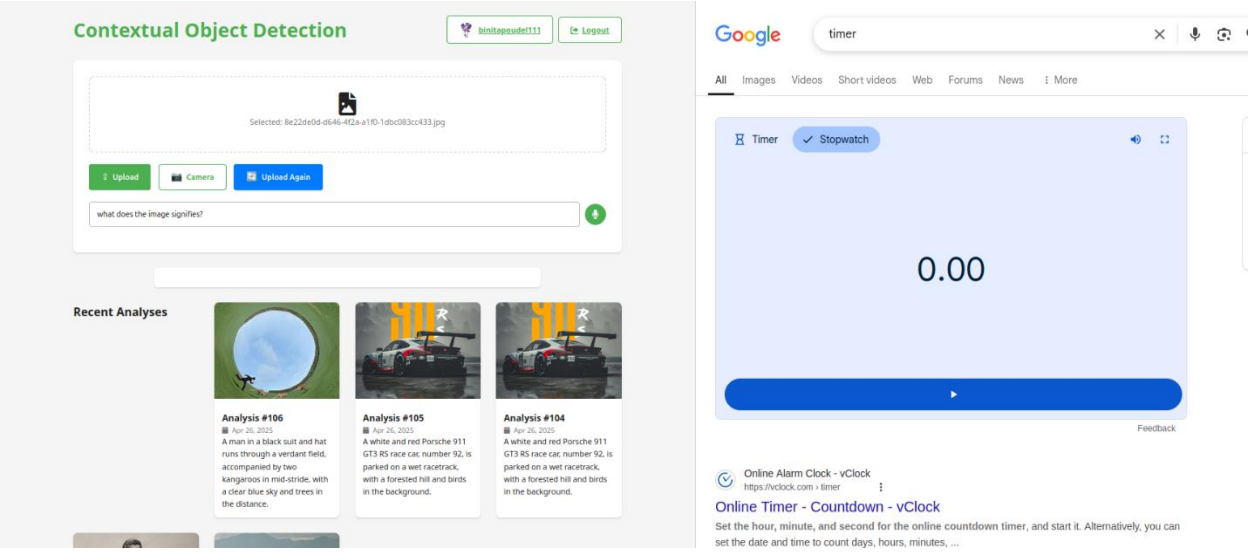


Figure 90: Uploading image with visual query

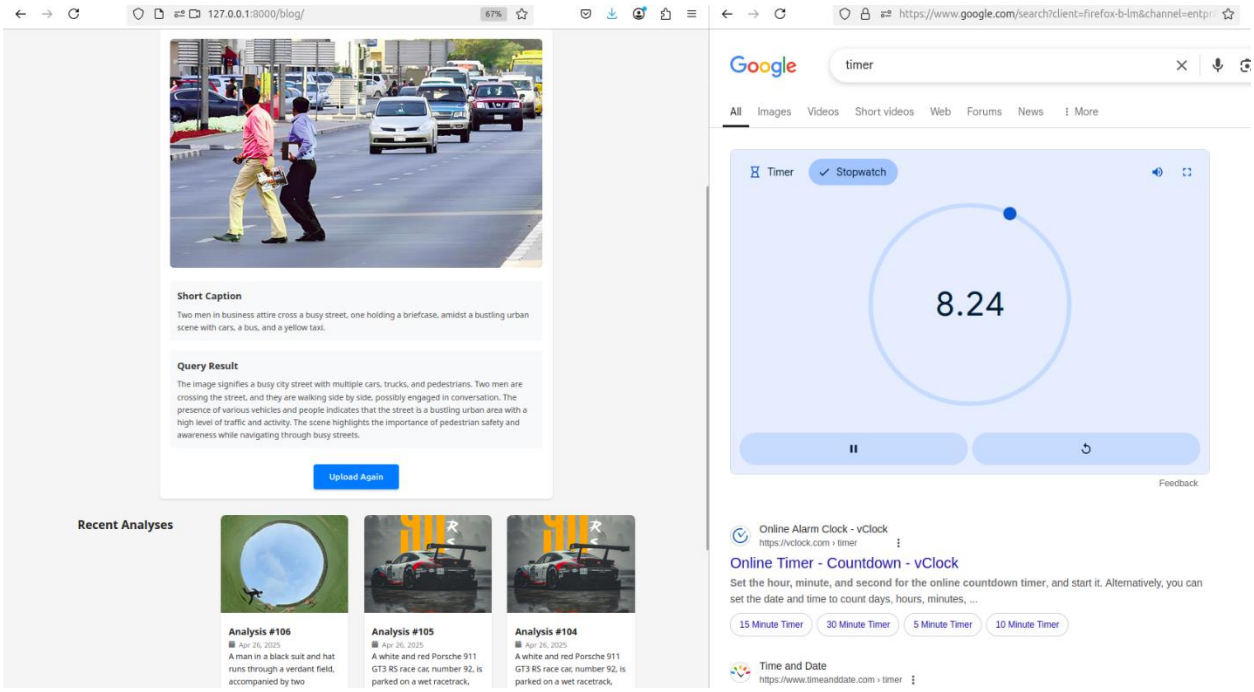


Figure 91: Context and query's answer within 8.24 seconds

4.2.22. Test Case 22: Terminate context generation.

Objectives	To terminate the context generation process.
Action	User should click the terminate button to stop the context generation process for the particular instance.
Expected Result	System should stop generating context for that instance.
Actual Result	System efficiently stopped the context generation.
Conclusion	Test passed.

Table 26: Test 22: Terminate context generation

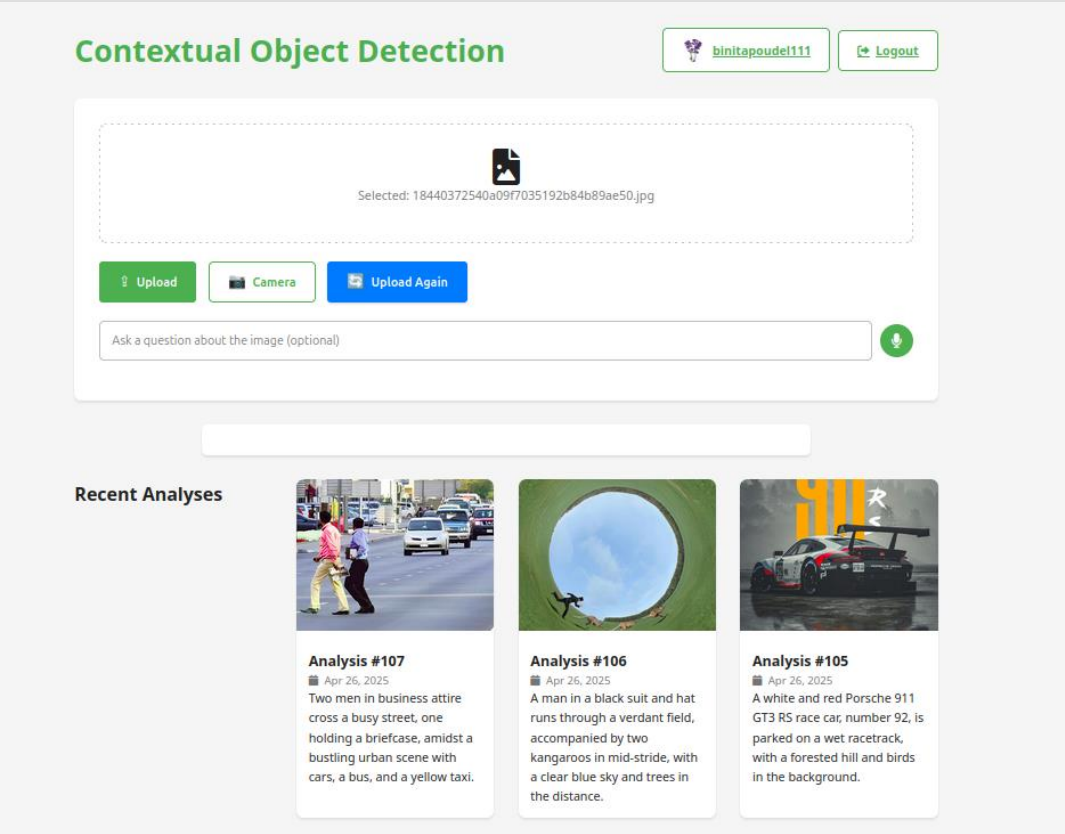


Figure 92: Uploading image for context

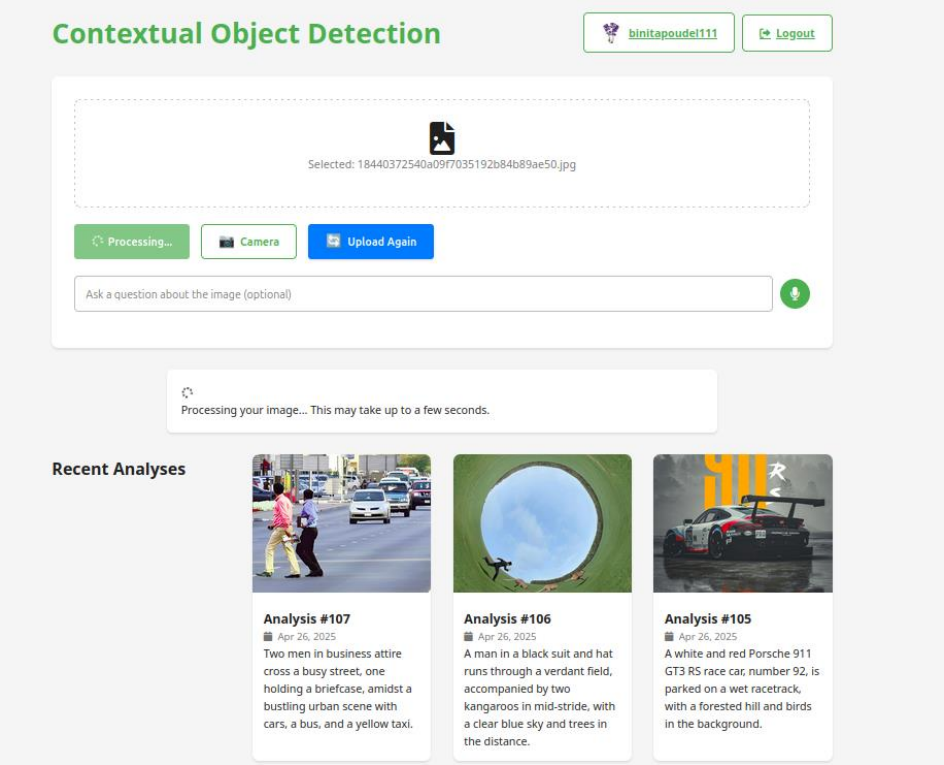


Figure 93: Processing the image

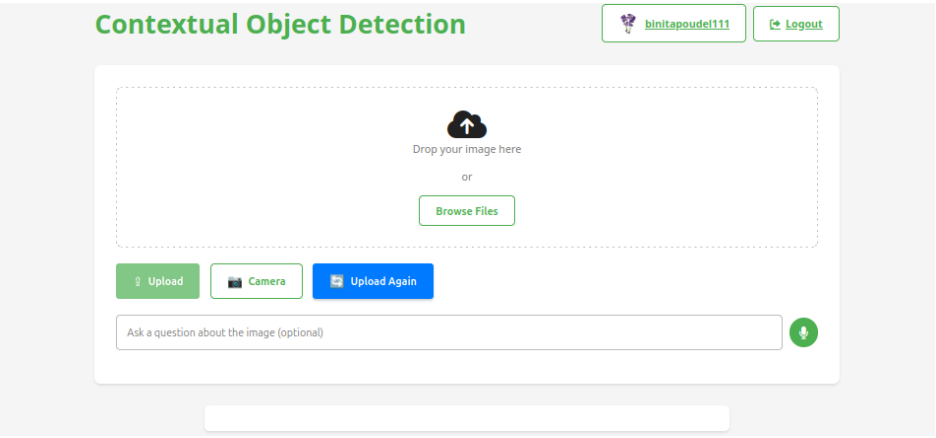


Figure 94: Terminating the generation process

4.2.23. Test Case 23: Create, read/view, update and delete (CRUD) operations for registered users in admin dashboard.

Objectives	To check the CRUD operations for registered user's details in admin dashboard.
Action	New user details are created, existing details were read, updated and deleted from admin dashboard and the database.
Expected Result	The system should allow the admin to perform CRUD operations in registered user's data.
Actual Result	Admin was successfully able to perform CRUD operations in the user's details from admin dashboard.
Conclusion	Test passed.

Table 27: Test 23: CRUD operations in user details by admin

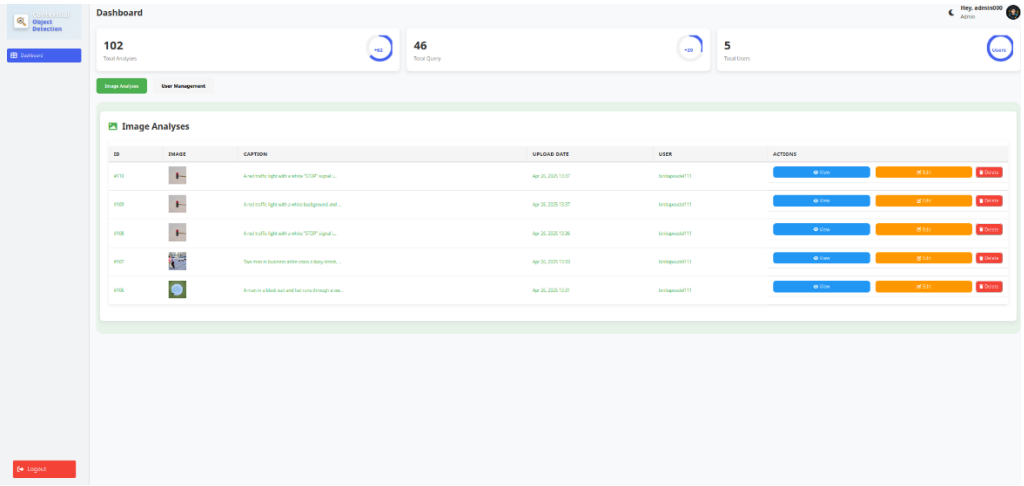


Figure 95: Admin dashboard

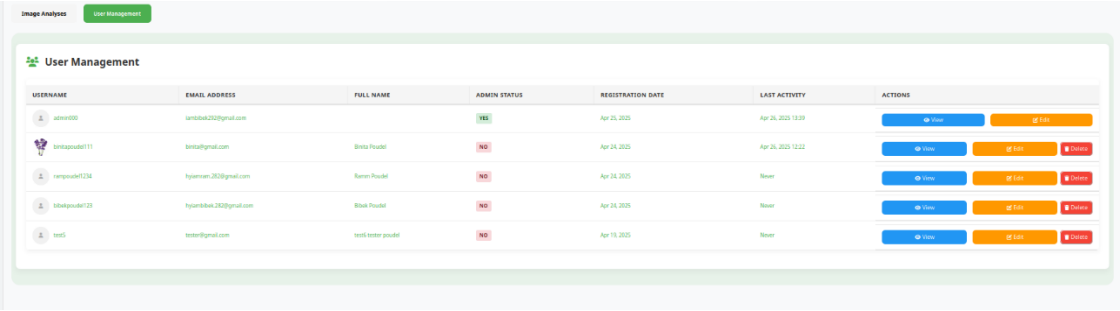


Figure 96: User's data for CRUD operations

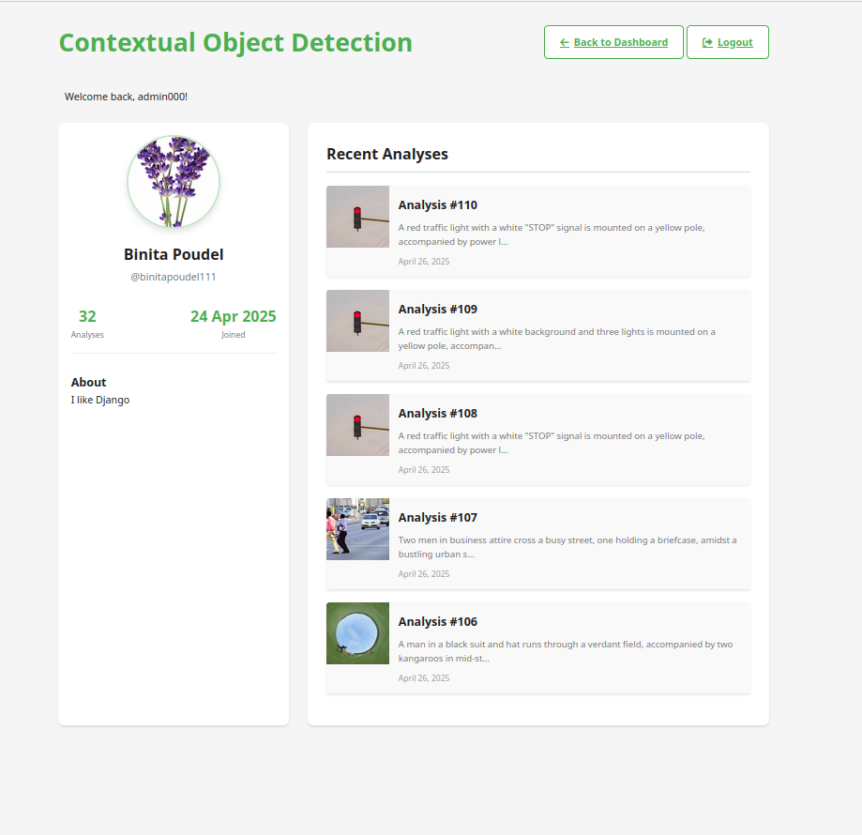


Figure 97: Viewing and reading user's information

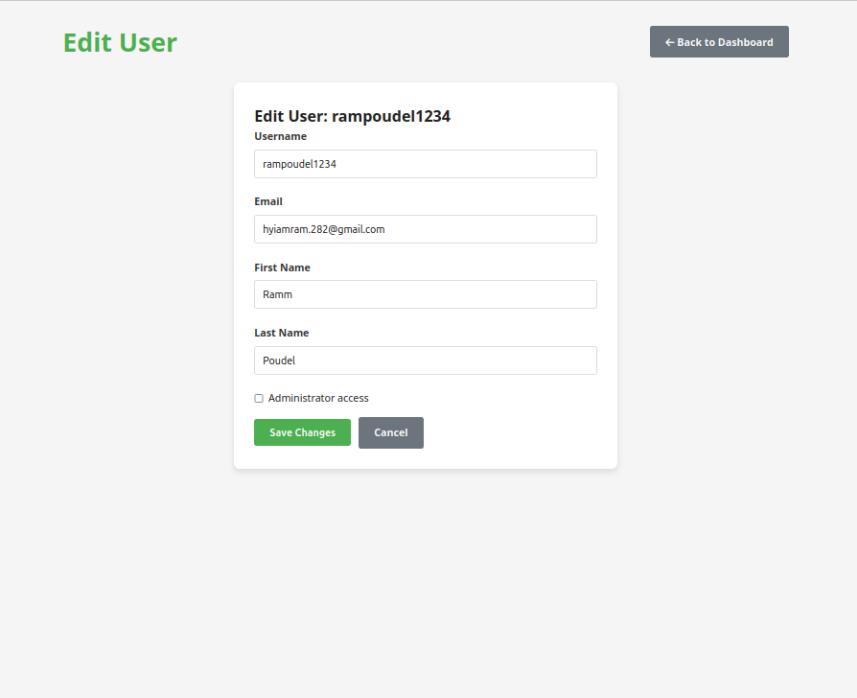


Figure 98: Updating the user's information

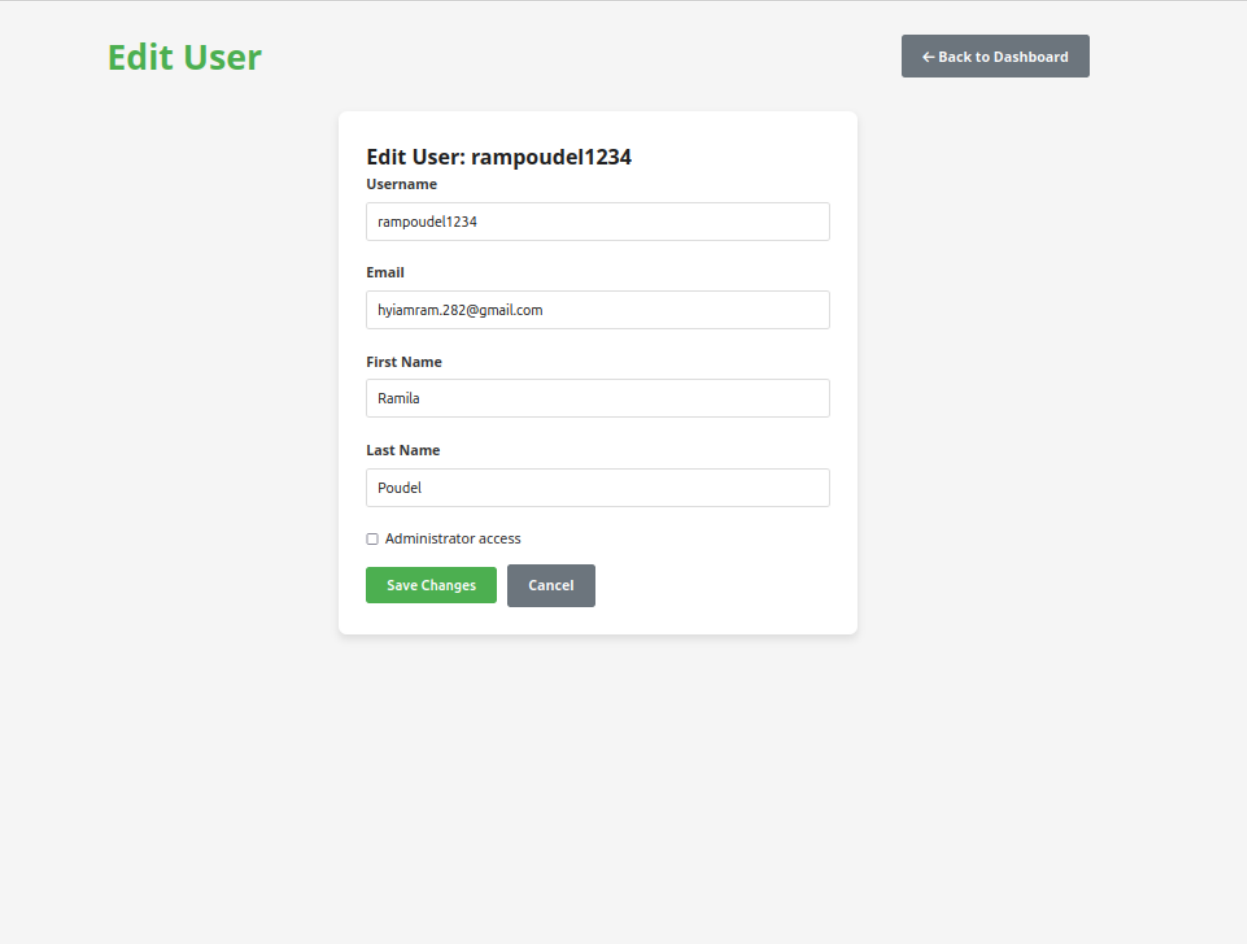


Figure 99: Updating user's name with new name

Figure 100: User's data updated successfully

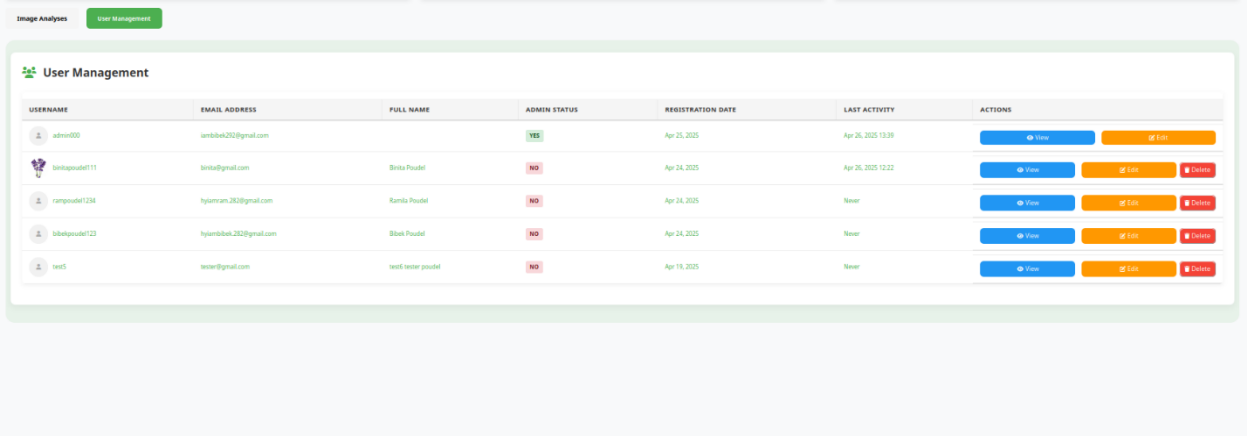


Figure 101: Before deleting the user

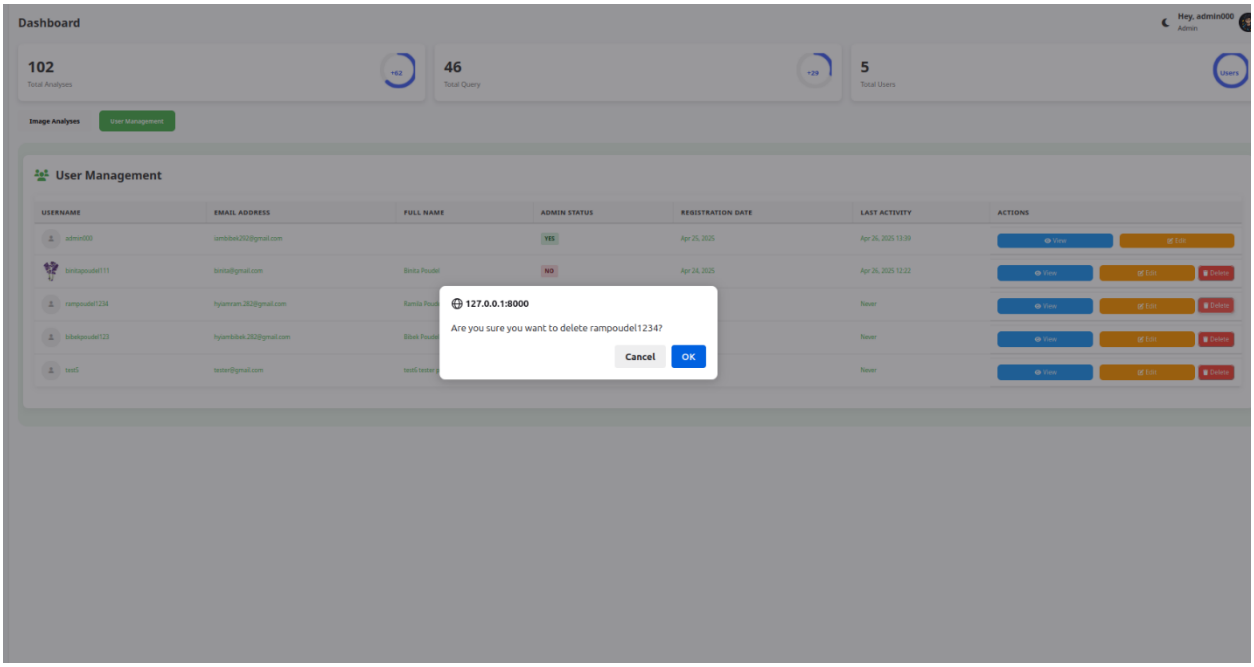


Figure 102: Deleting the user

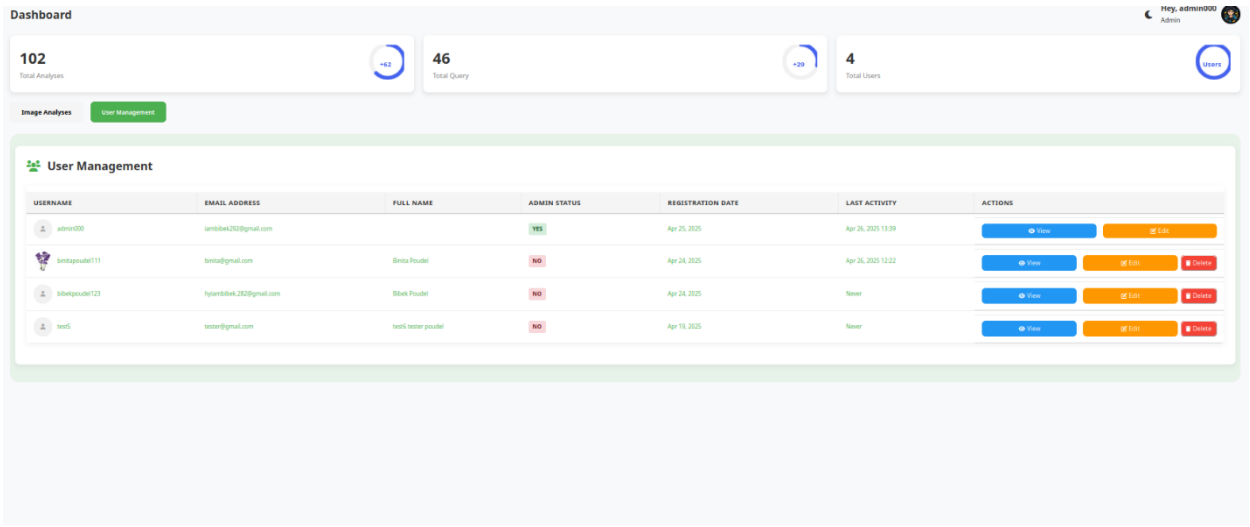


Figure 103: Deleted the user successfully

4.2.24. Test Case 24: Create, read/view, update and delete (CRUD) operations for generated context and images in admin dashboard.

Objectives	To check the CRUD operations implementations in processed images for context generation.
Action	Admin updated existing generated context, and deleted them.
Expected Result	The details about the processed images which were updated and deleted should be change accordingly with permanent changes in database.
Actual Result	The admin was able to view, update and deleted the generated contexts along with images.
Conclusion	Test passed.

Table 28: CRUD operations on analysed images and context by admin

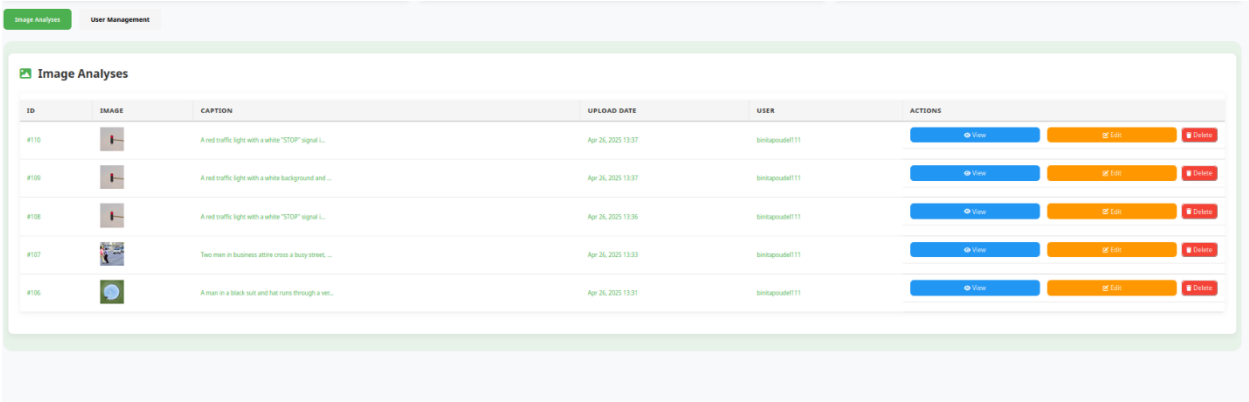


Figure 104: Recently analysed images in admin dashboard

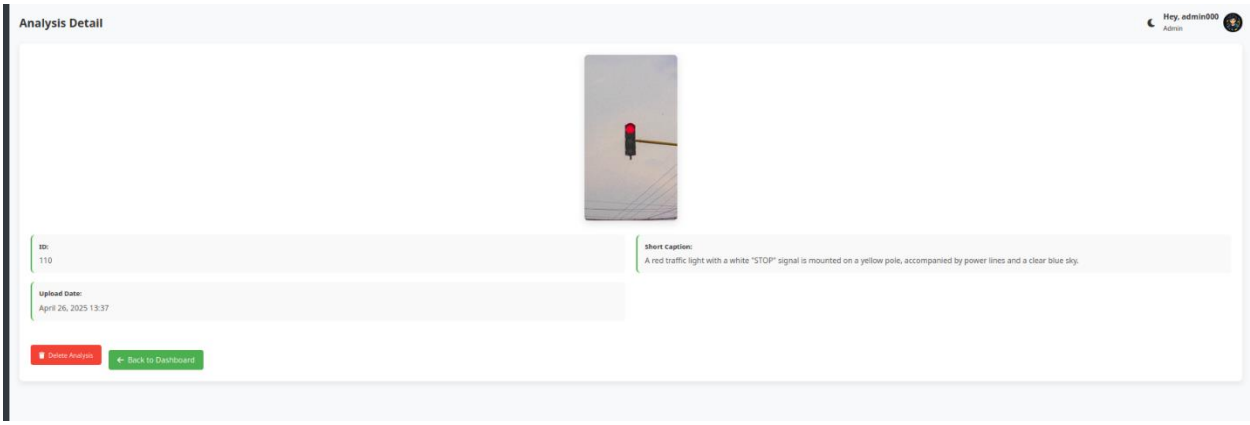


Figure 105: Viewing/reading the analysed image's information

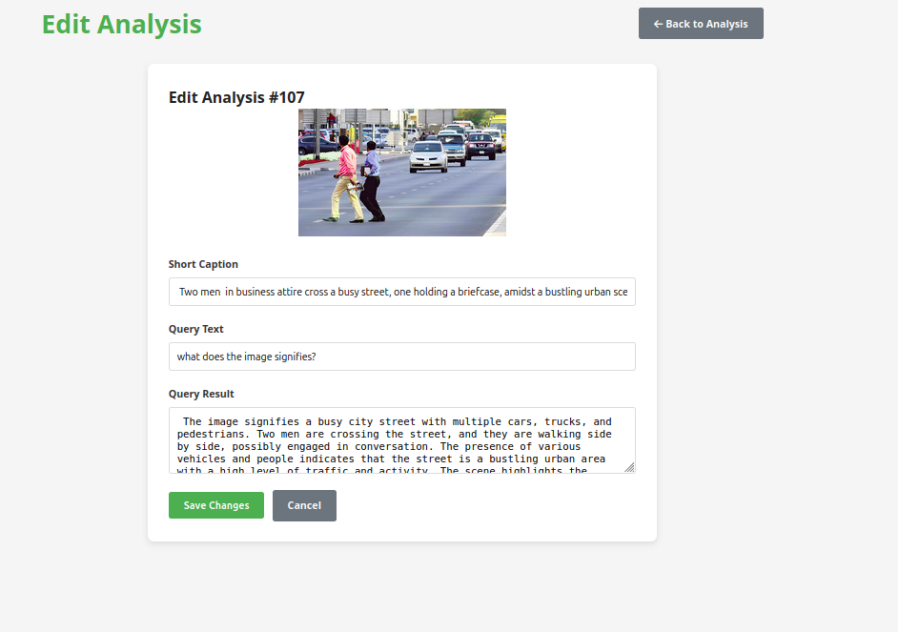


Figure 106:Updating the analysis and context

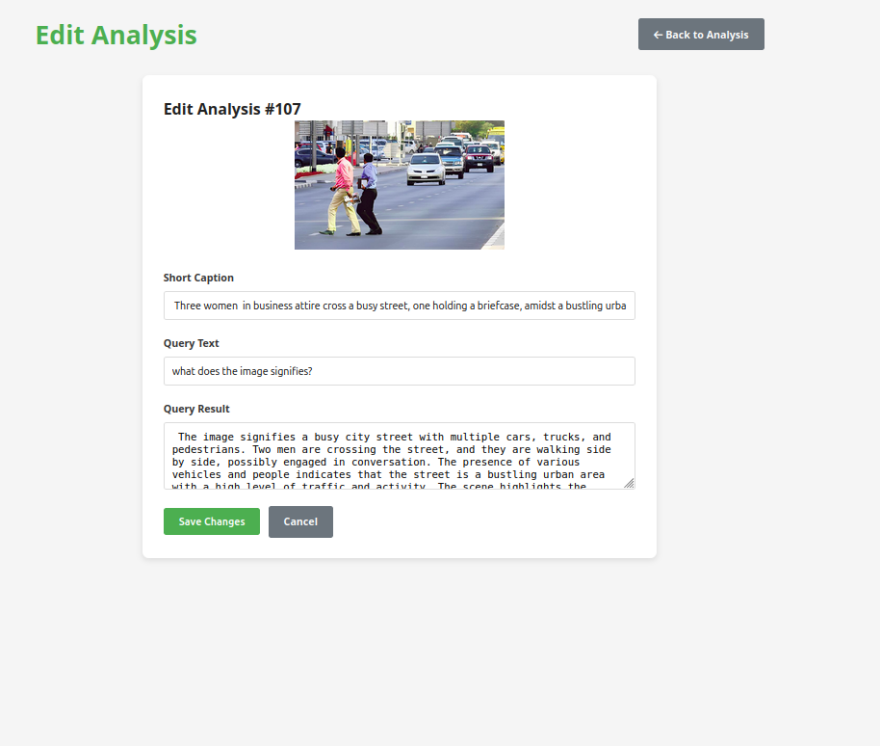


Figure 107: Updating the context with different text

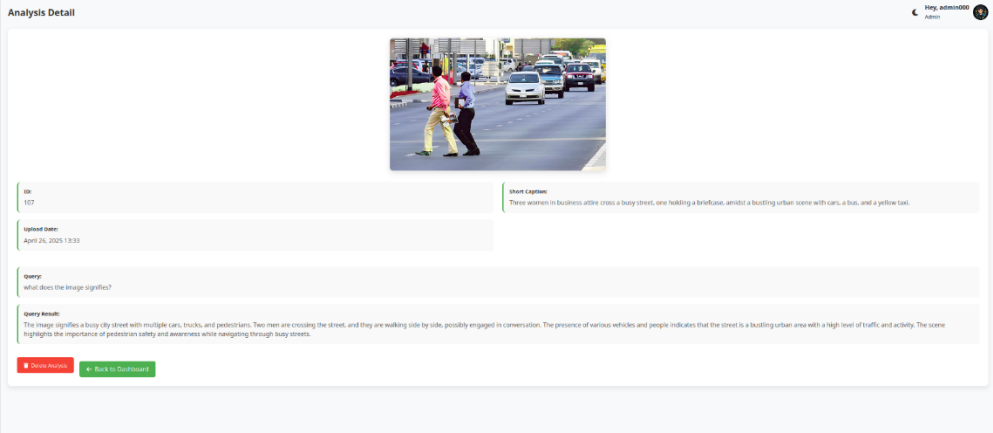


Figure 108: Context updated successfully

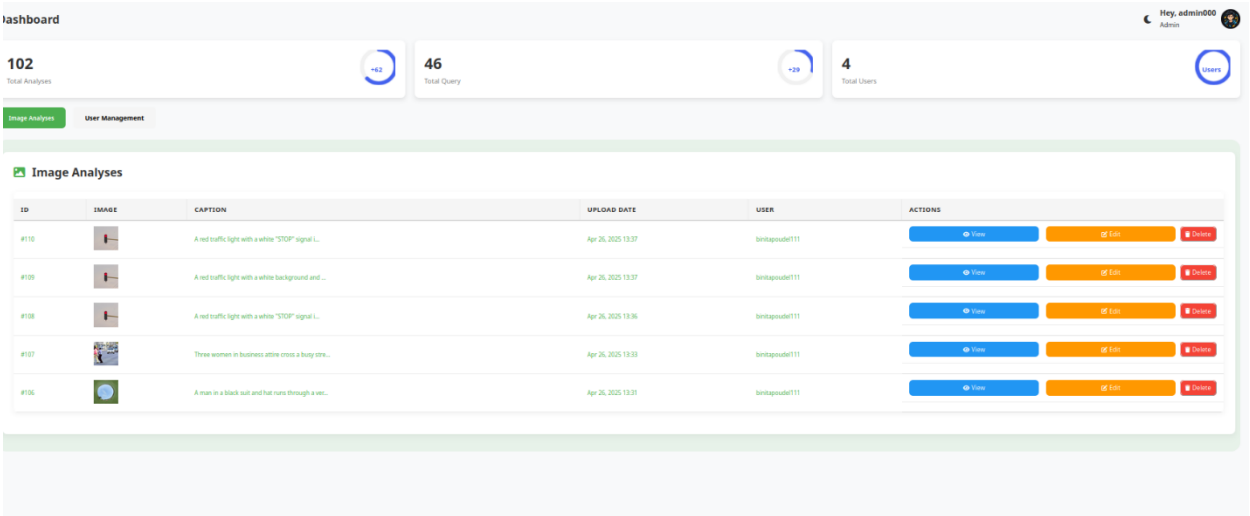


Figure 109: Before deleting the context and image

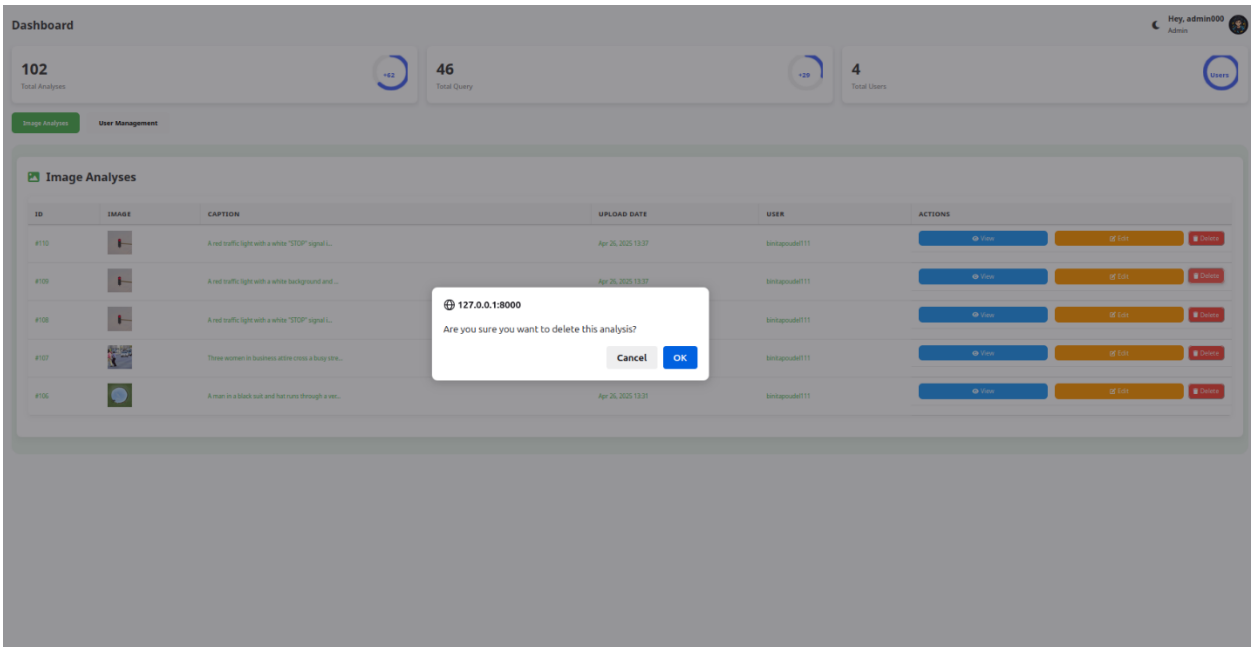


Figure 110: Deleting the context and the image

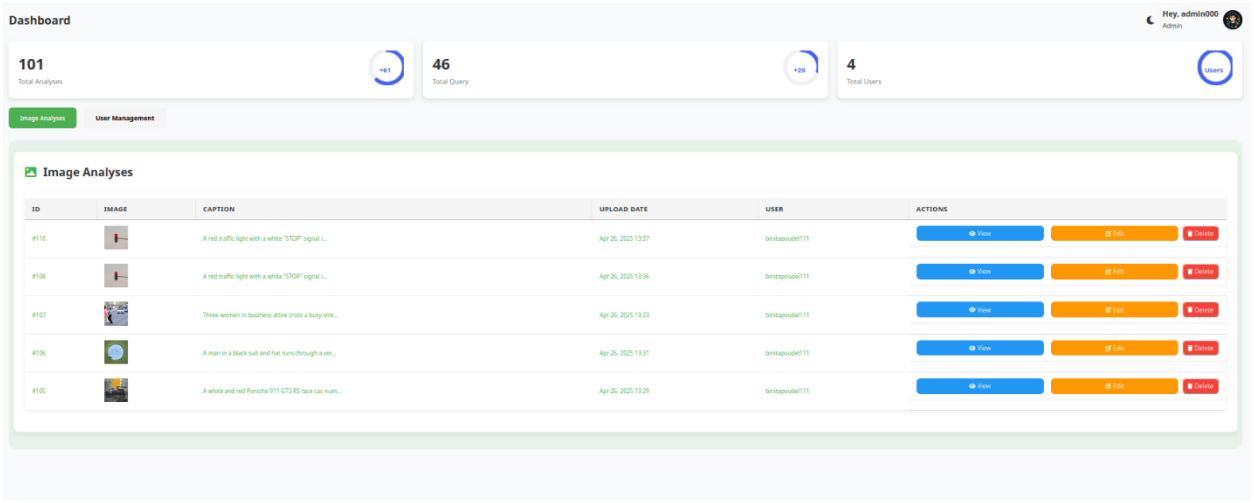


Figure 111: Successfully deleted the context of the image

4.3. System Testing, Testing Plan

System testing is a level of software testing where the complete integrated system is tested to evaluate its compliance with specified requirements, ensuring the application functions correctly as a whole (Das, 2024). System testing covers the end-to-end functions of a system, and thus it provides reliability to the system. This testing keeps the new and previous functionalities in a single system to help the user understand the benefits of the newly added features (Das, 2024).

Test Case	**Objective**
1	To test the complete journey of the user in system.
2	End-to-End Processing Pipeline Test
3	Cross-Feature Interaction Test
4	Cross Device Configuration User Experience Test
5	Image processing Pipeline and different image file format
6	Running full system on Linux machine
7	To test the context generated from the model on benchmark dataset as ground truth
8	To test admin's ability to view admin dashboard and access exclusive features.

Table 29: System testing plan

4.3.1. Test 1: Complete User Journey Test

Objectives	To verify the complete user journey from registration to multiple analyses in context generation page.
Action	i. Register new account with relevant credentials ii. Login with correct credentials iii. Upload an image iv. View Analysis v. Update the profile picture vi. Update the bio vii. Again, upload the image and generate the analysis viii. Log out of the system.
Expected Result	All steps should complete successfully with appropriate transitions, data persistence between sessions.
Actual Result	All steps completed successfully with data persistence between the sessions.
Conclusion	Test passed.

Table 30: Sys Test 1: To verify the complete user journey in system

Screenshots

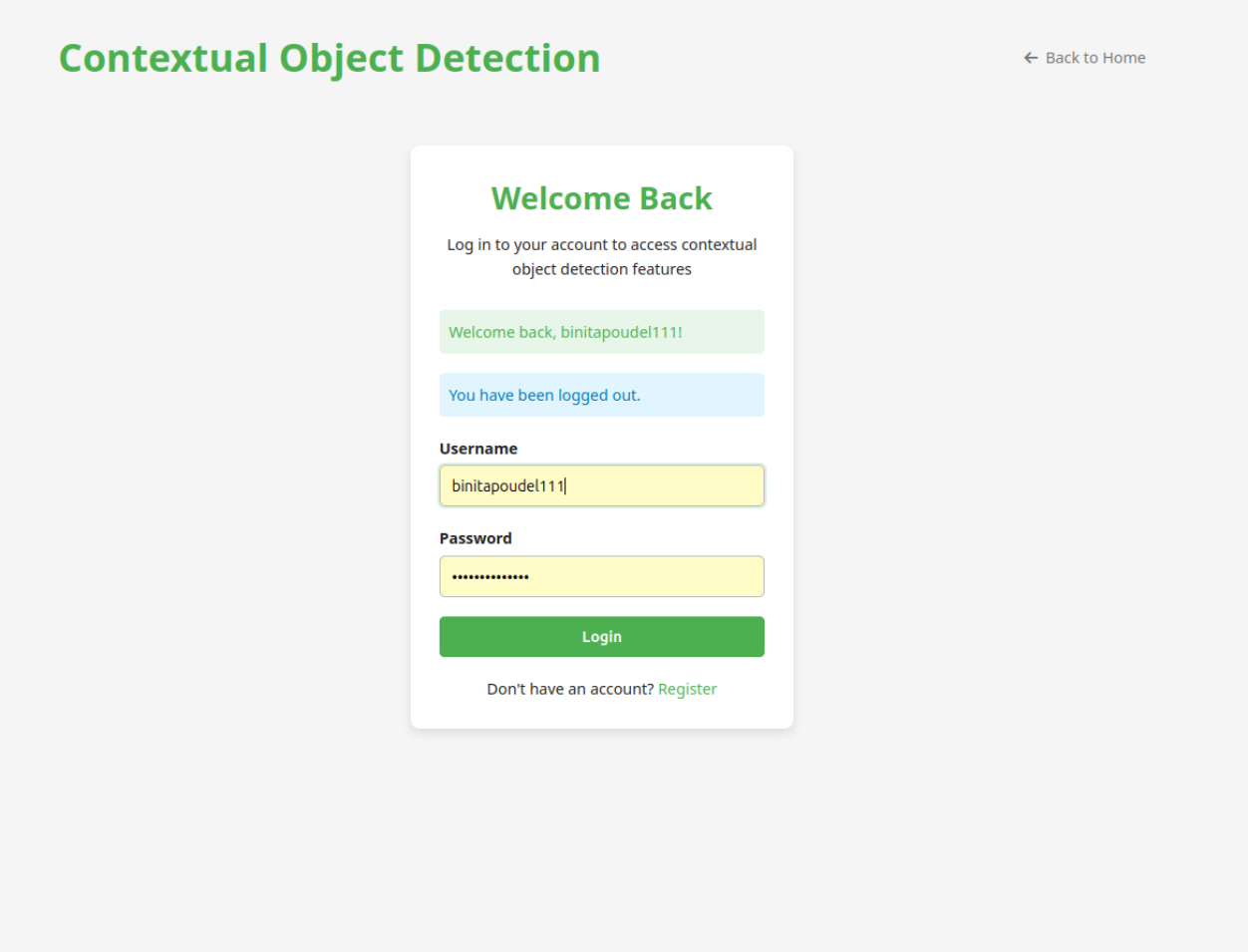


Figure 112: Log in to the system with credentials registered

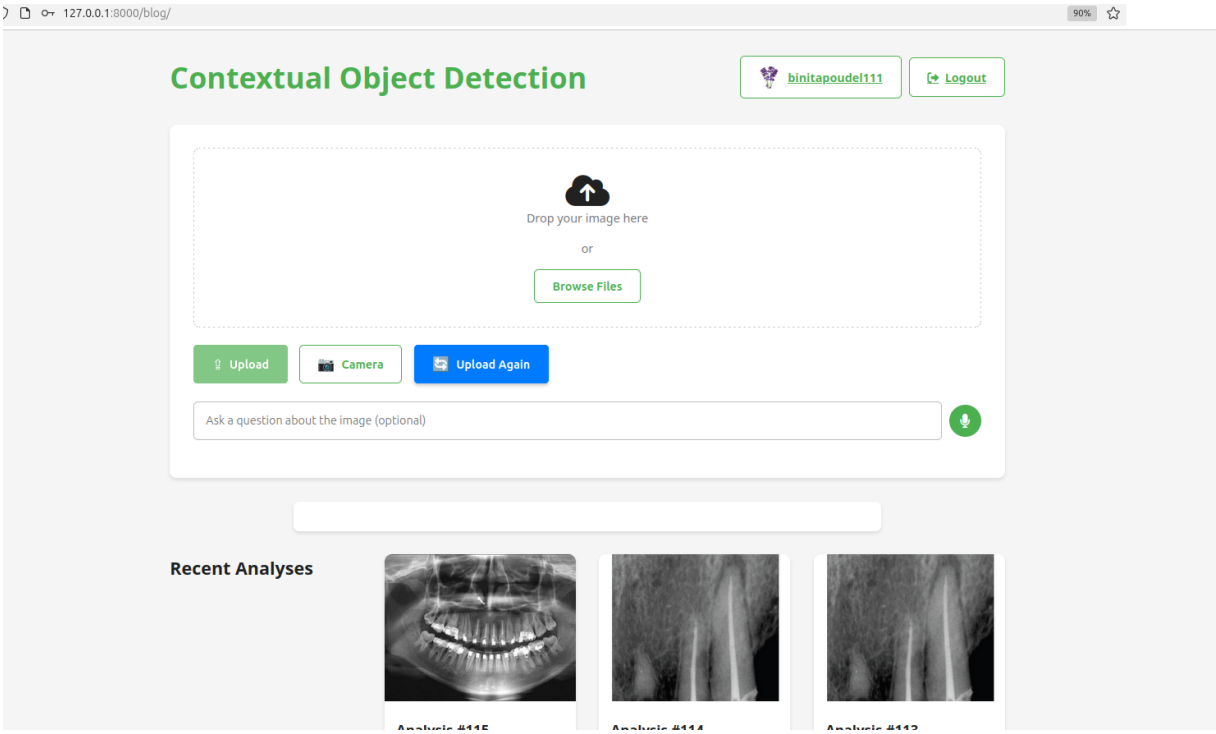


Figure 113: Greeted with context gneration page

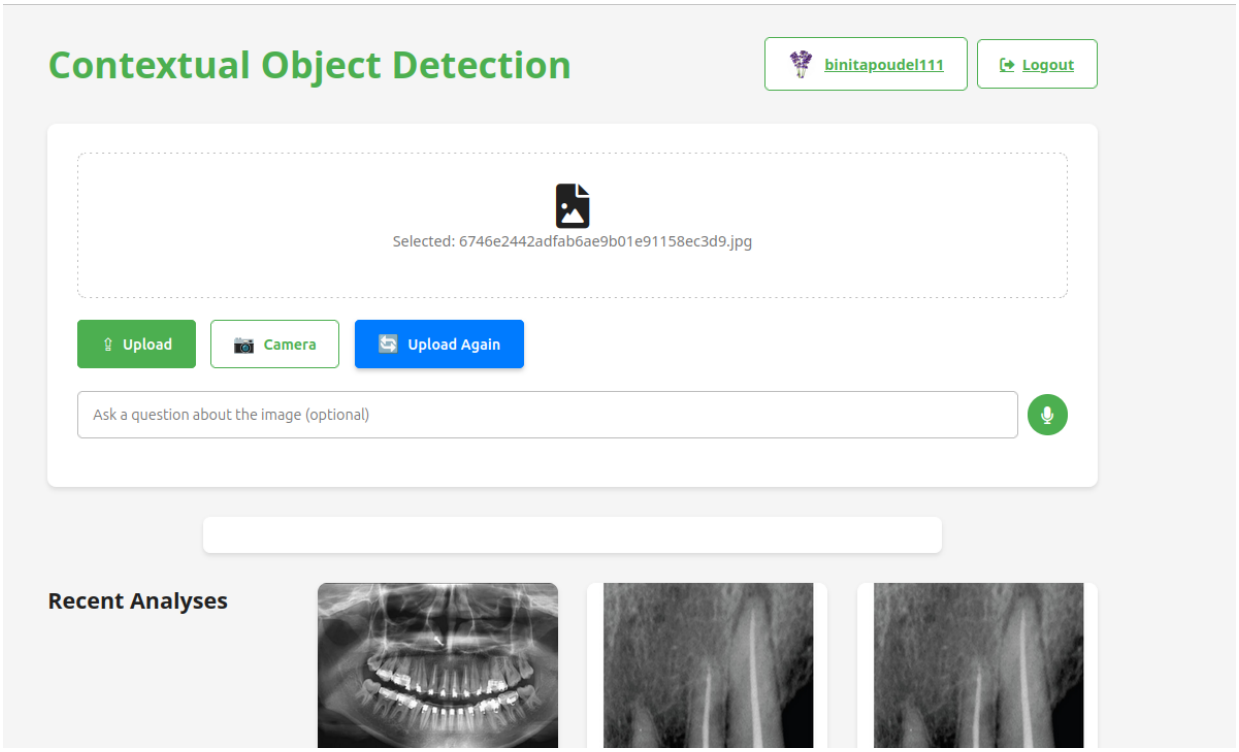


Figure 114: Uploading image

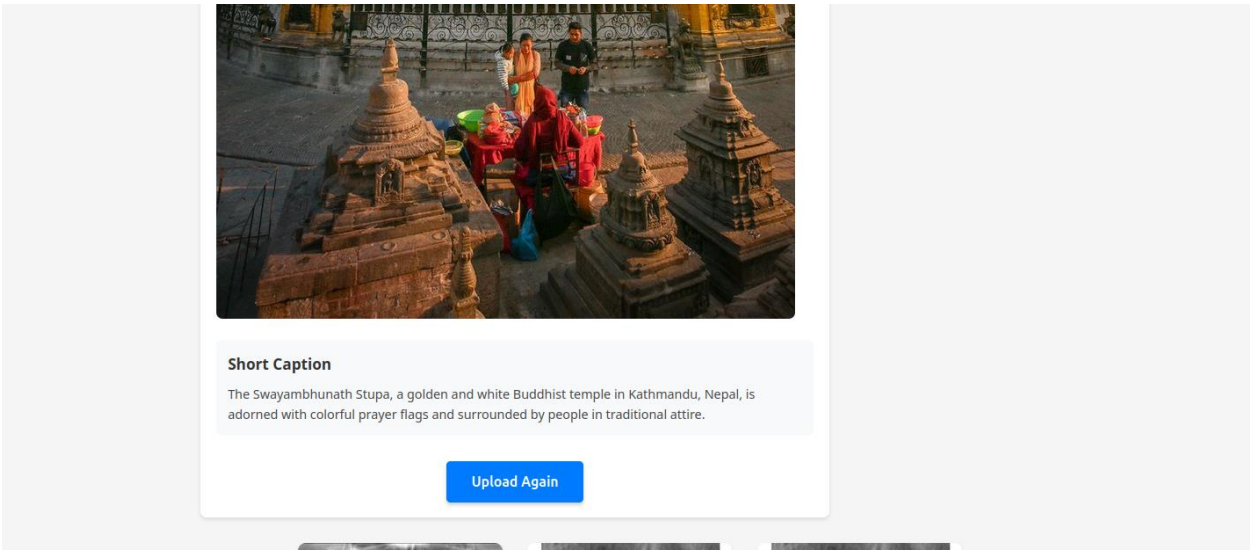


Figure 115: Context generation

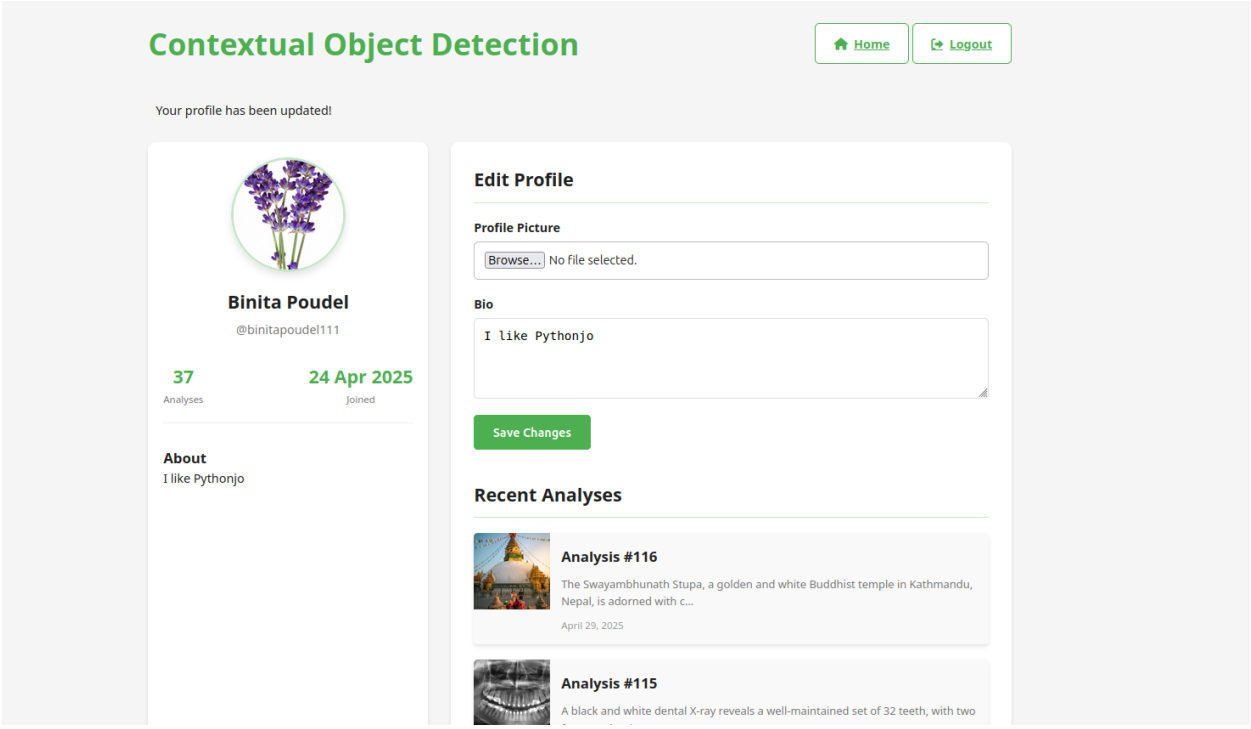


Figure 116: Visiting profile

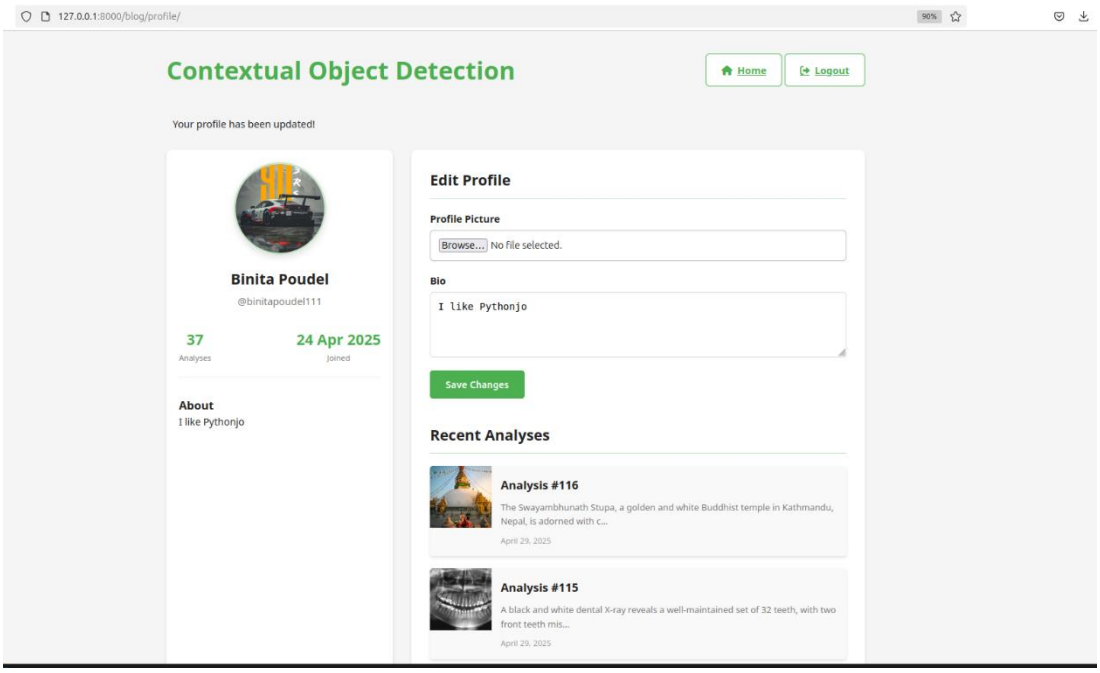


Figure 117: Updating profile picture

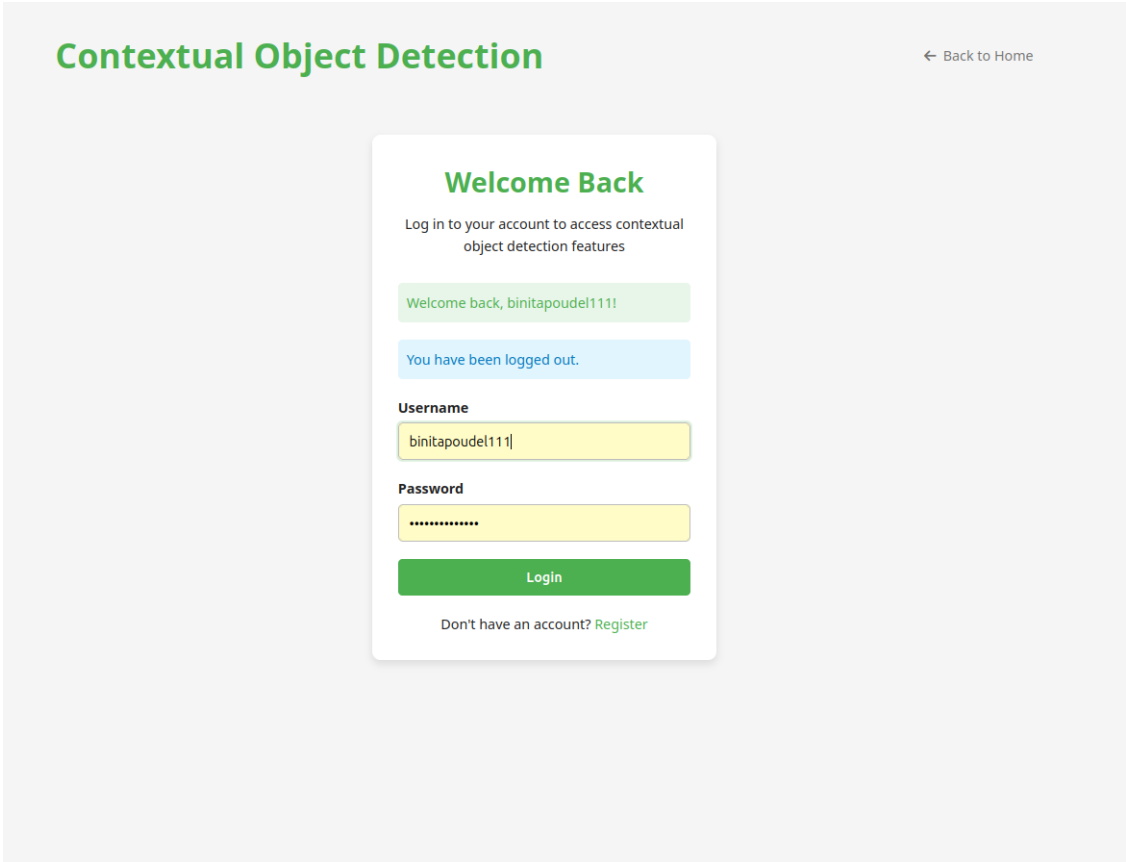


Figure 118: Logout from system

4.3.2. Test 2: End-to-End Processing Pipeline Test

Objectives	To verify entire image processing pipeline from upload through Redis queue to final results.
Action	i. Upload 5 different image (landscape, portrait, text-heavy, object-heavy, low-light) and monitor the complete processing pipeline.
Expected Result	The pipeline should efficiently upload, queue, process and provide results to the images.
Actual Result	The pipeline successfully performs the image processing task efficiently.
Conclusion	Test passed.

Table 31: SysTest 2: End-to-End processing pipeline Test

Screenshots

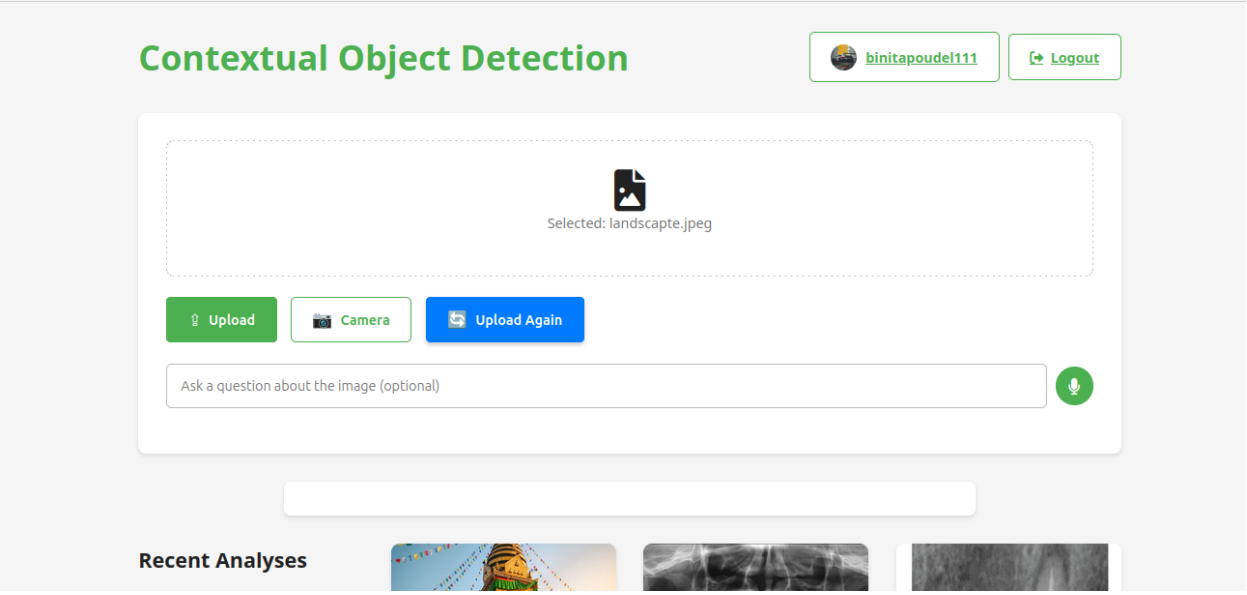


Figure 119: Uploading landscape image

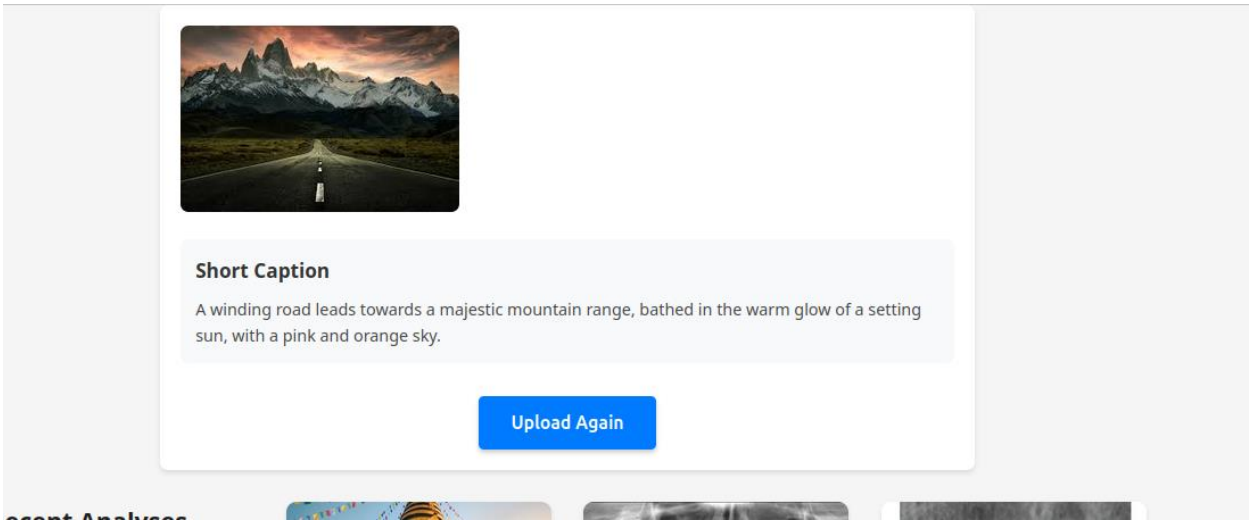


Figure 120: Context for landscape image

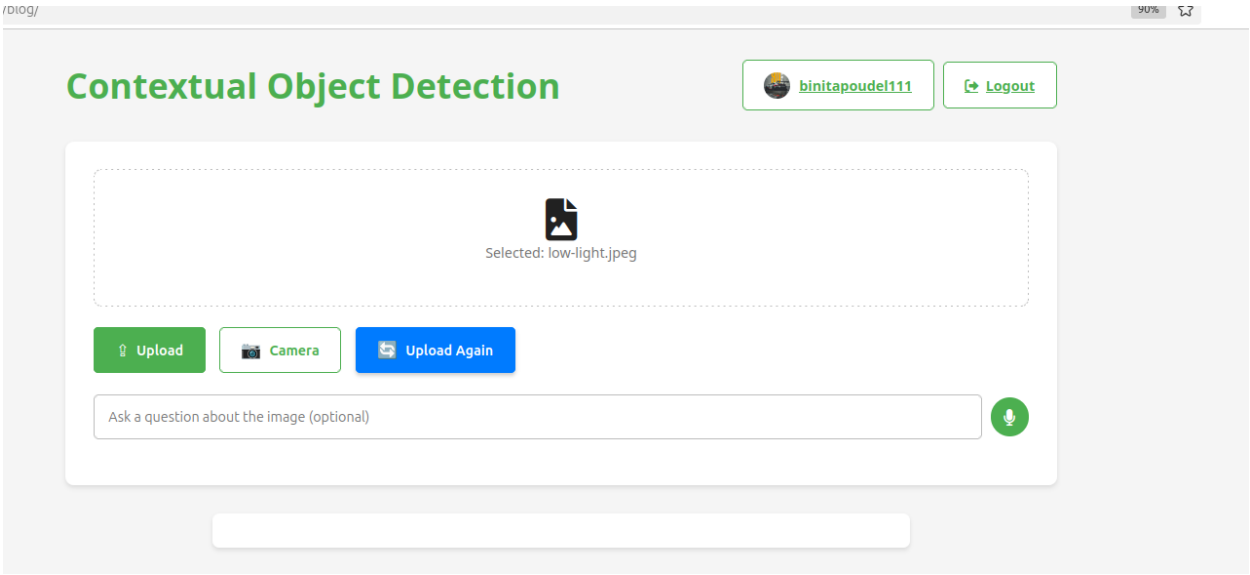


Figure 121: Uploading lowlight image

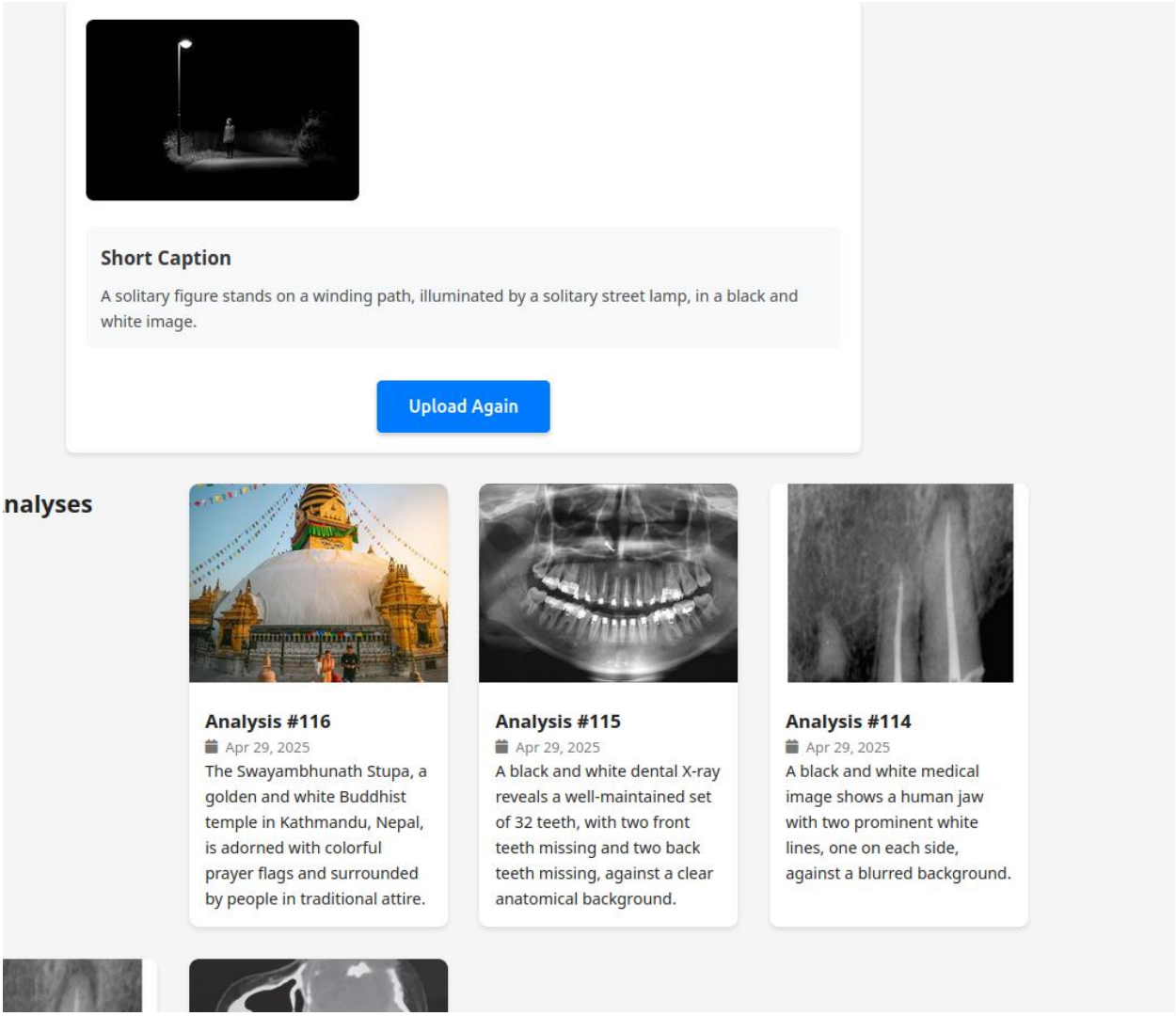


Figure 122: Generated context for low light image

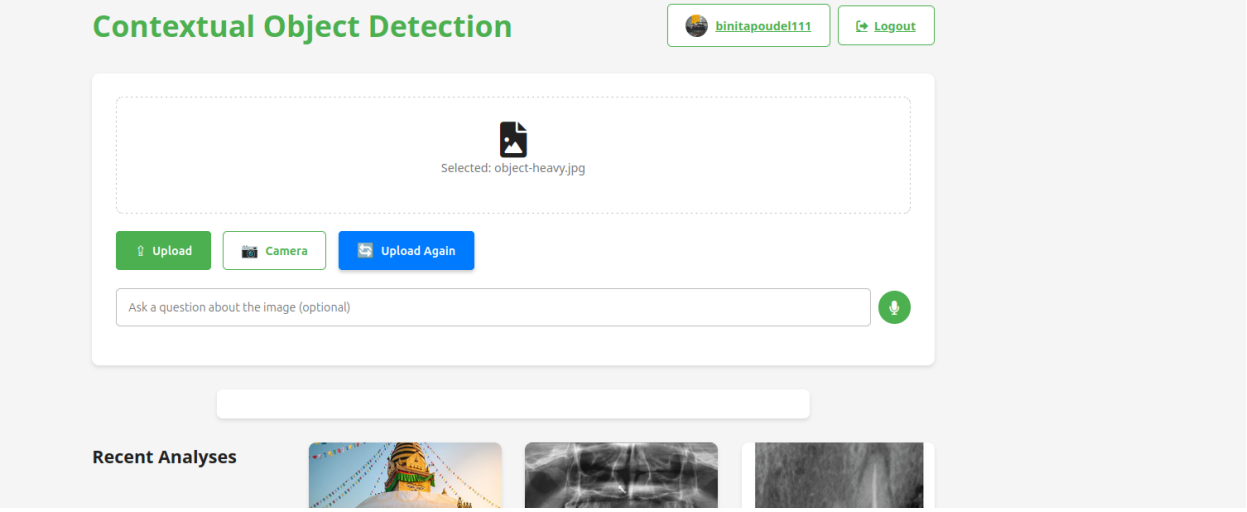


Figure 123: Uploading object heavy image

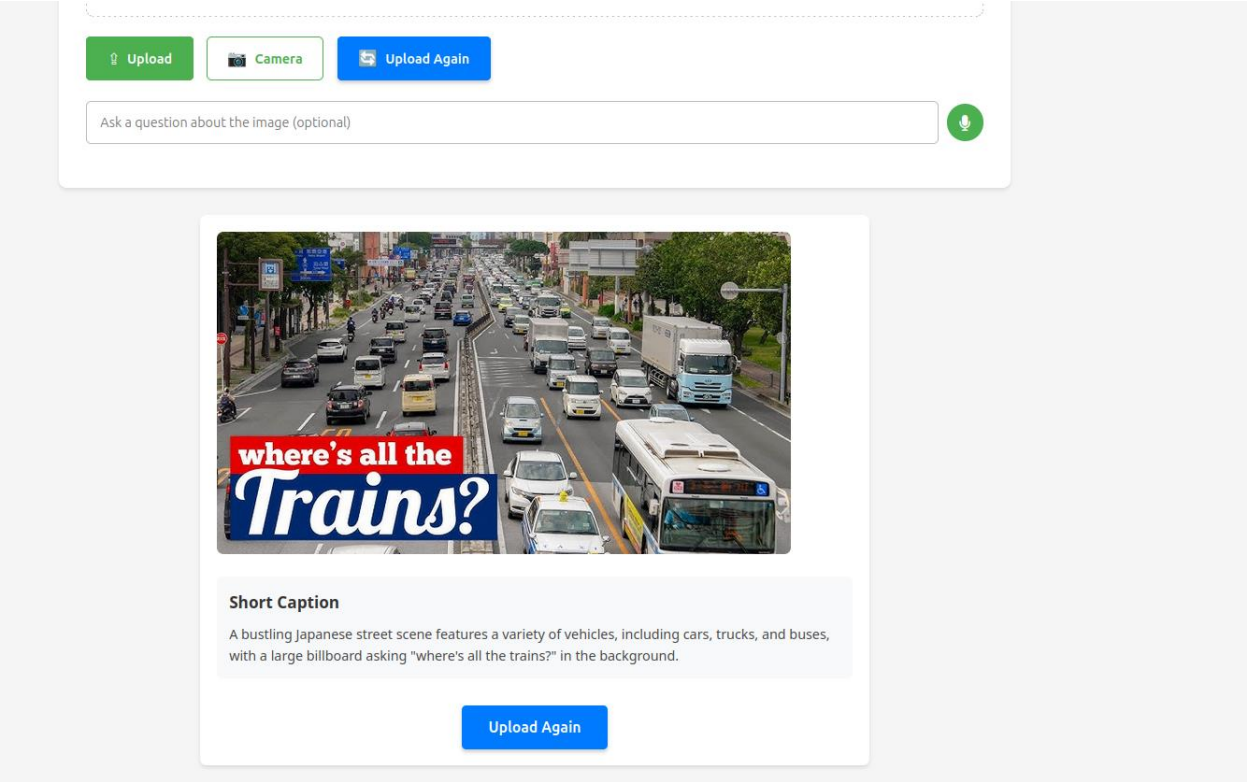


Figure 124: Context for object heavy image

4.3.3. Test 3: Cross-Feature Interaction Test

Objectives	To verify the interactions between different system features.
Action	Test interaction between features: using speech-to-text to add query for image analysis.
Expected Result	The system should efficiently integrate two different feature to complete the user's operation of context generation.
Actual Result	System successfully integrated the features.
Conclusion	Test passed.

Table 32: SysTest 3: Cross-feature integration test

Screenshots

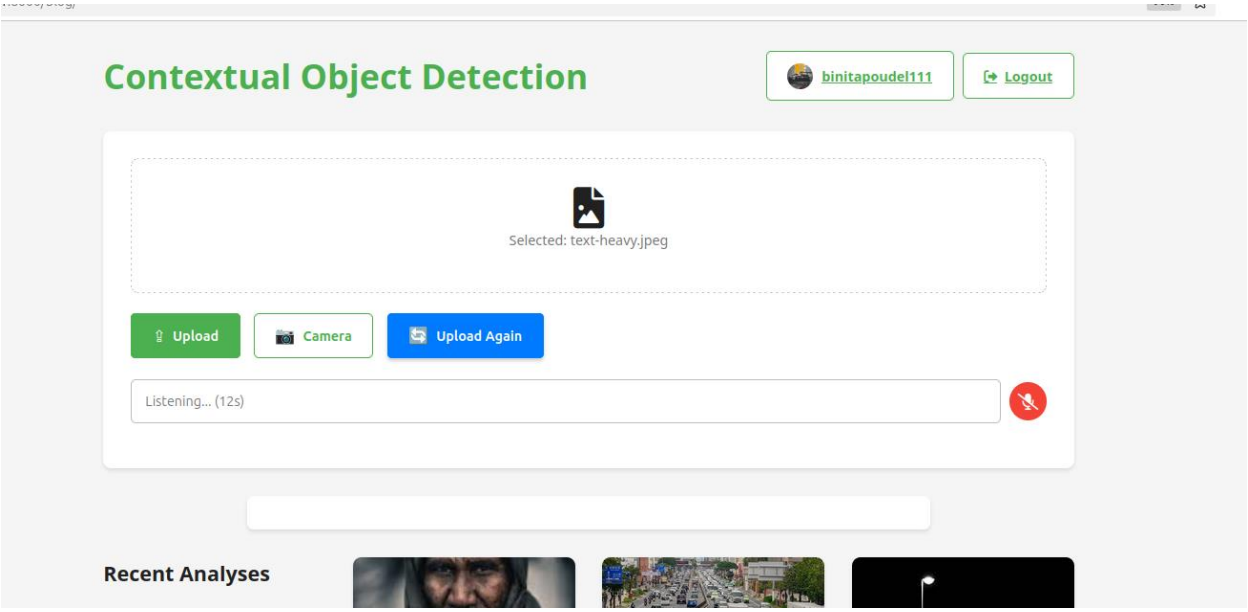


Figure 125: Uploading image for context

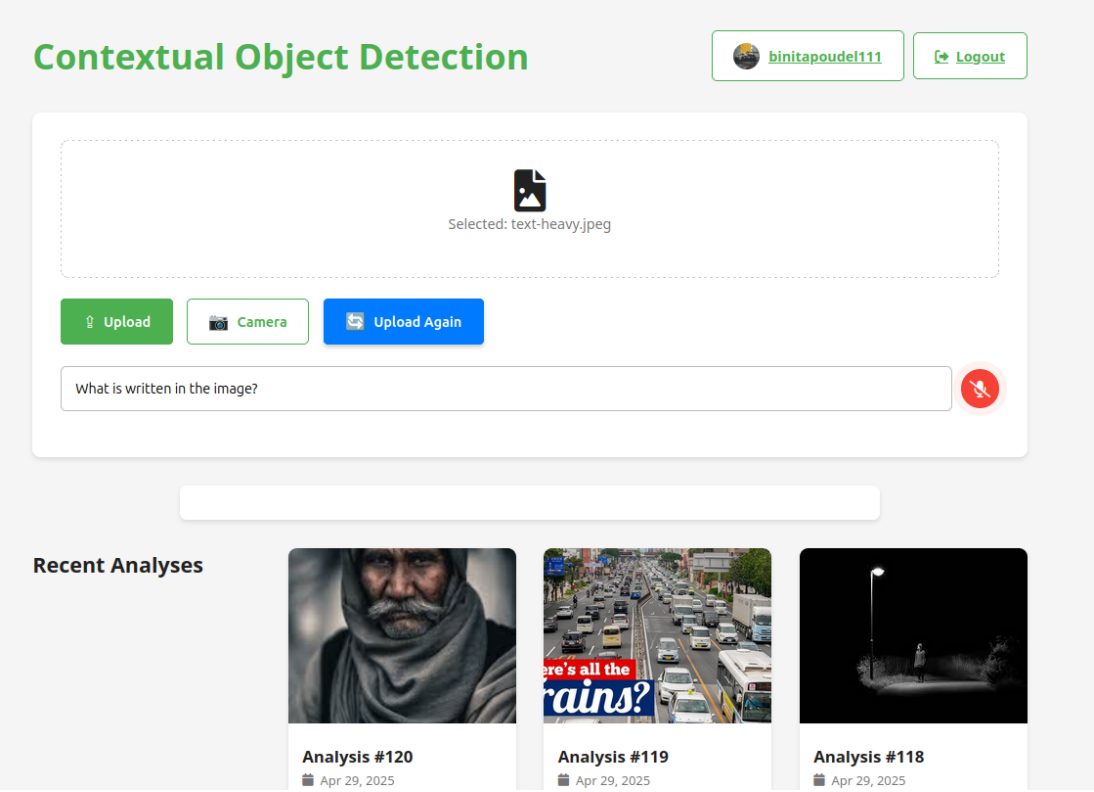


Figure 126: Speech to text for visual query to the image

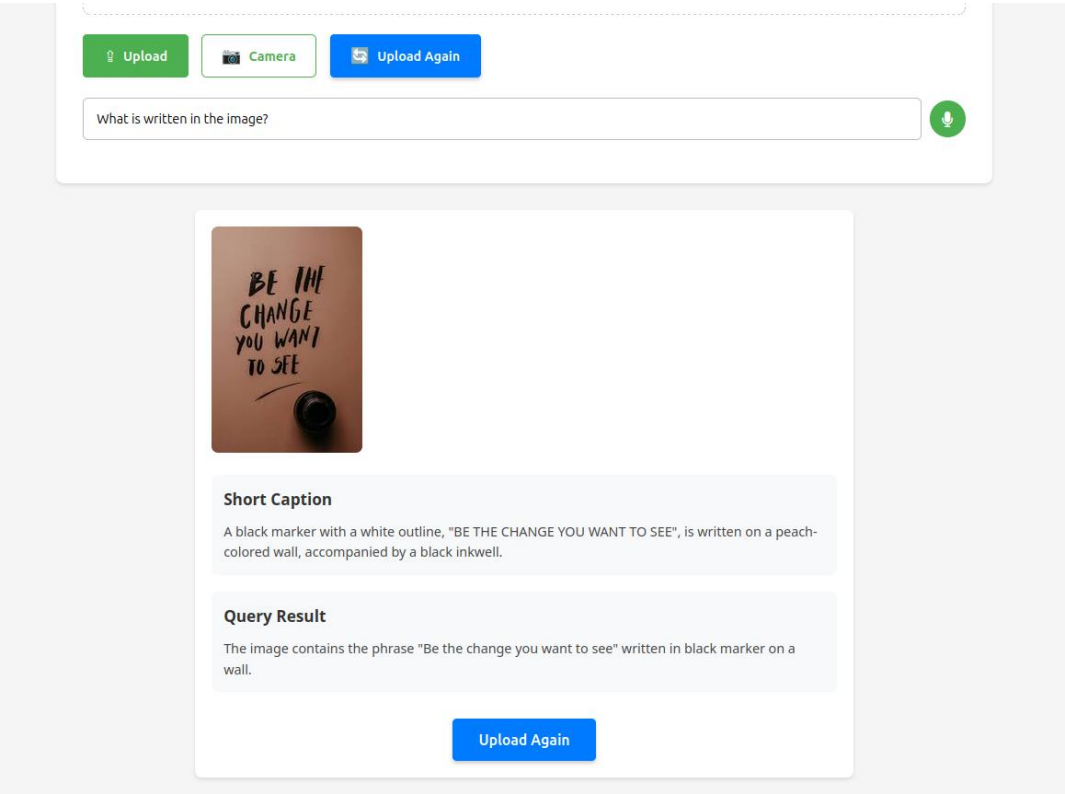


Figure 127: Context and visual query generated

4.3.4. Test 4: Cross Device User Experience Test

Objectives	To verify that the UI, functionality and data are maintained for all device configuration.
Action	i. Register a new user ii. Upload images and update the profile iii. Use Desktop configuration to log in with the created user and again perform the context generation and profile viewing. iv. Switched to Mobile configuration to login and performed the context generation and profile viewing. v. Finally, switched to tablet configuration to login and perform the context generation and profile viewing.
Expected Result	The system should function properly in all different devices configuration keeping the data persistence maintained.
Actual Result	The system successfully functioned on different device configurations.
Conclusion	Test passed.

Table 33: Cross device configuration test with UI

Screenshots

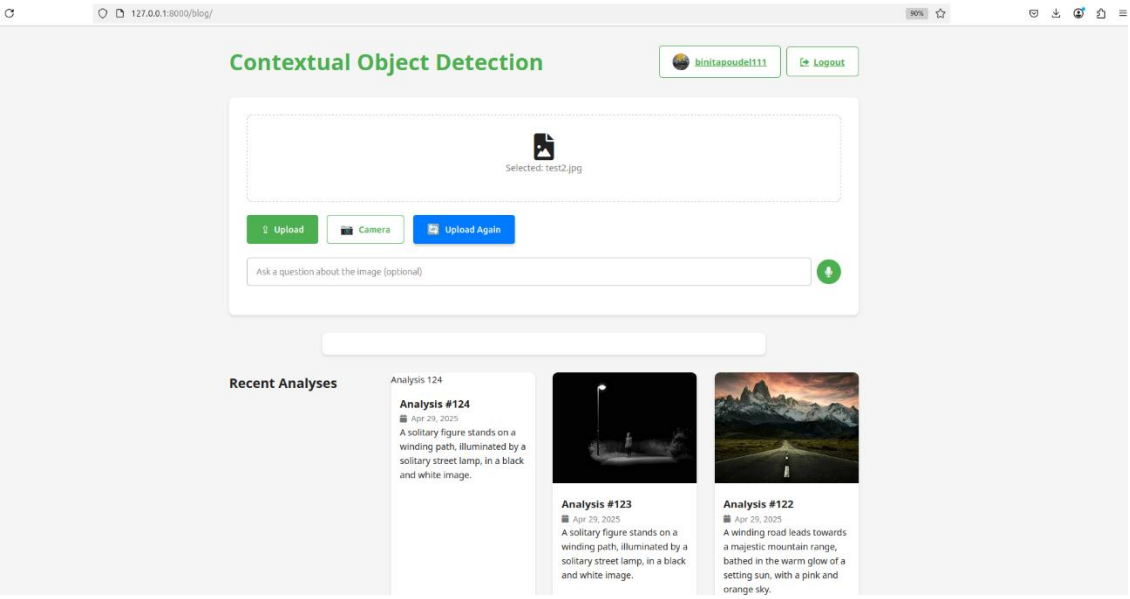


Figure 128: Main page in desktop configuration

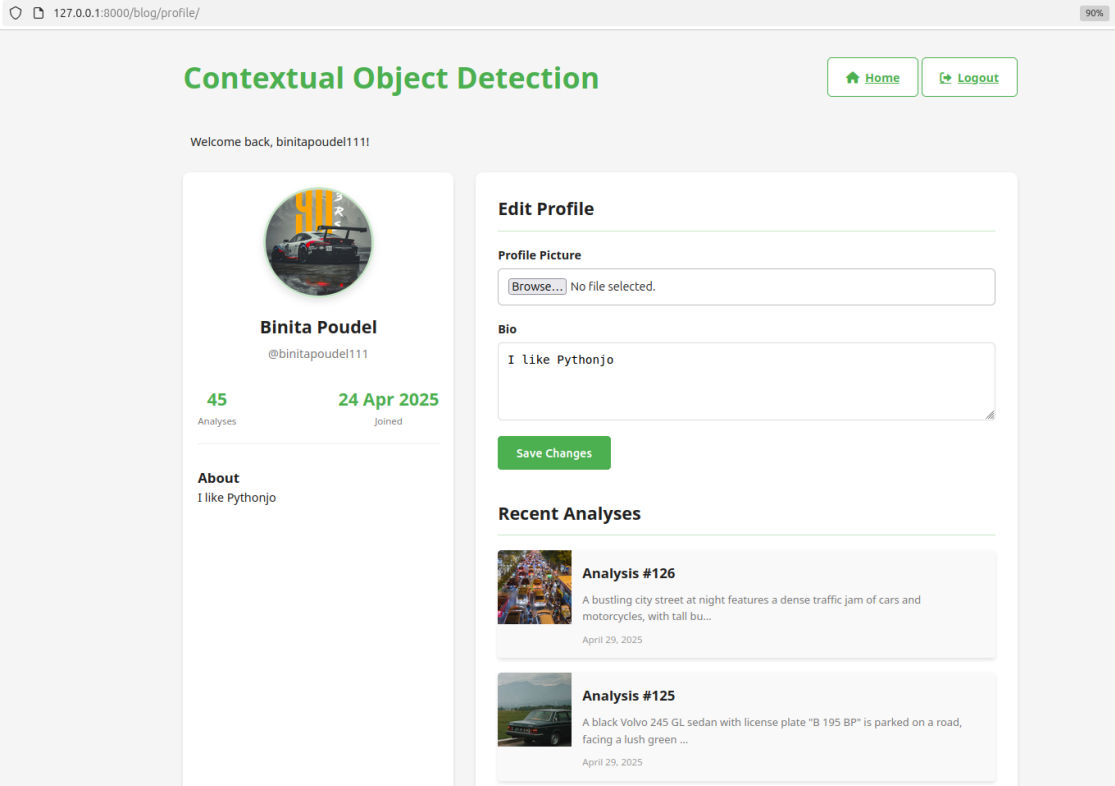


Figure 129: Profile in desktop configuration

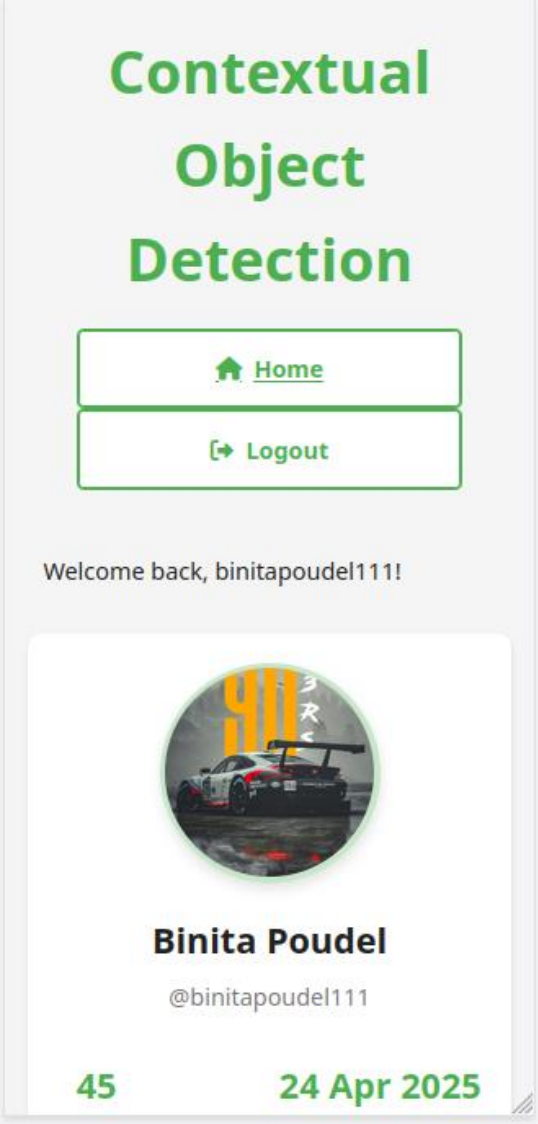


Figure 130: Profile view in Mobile configuration

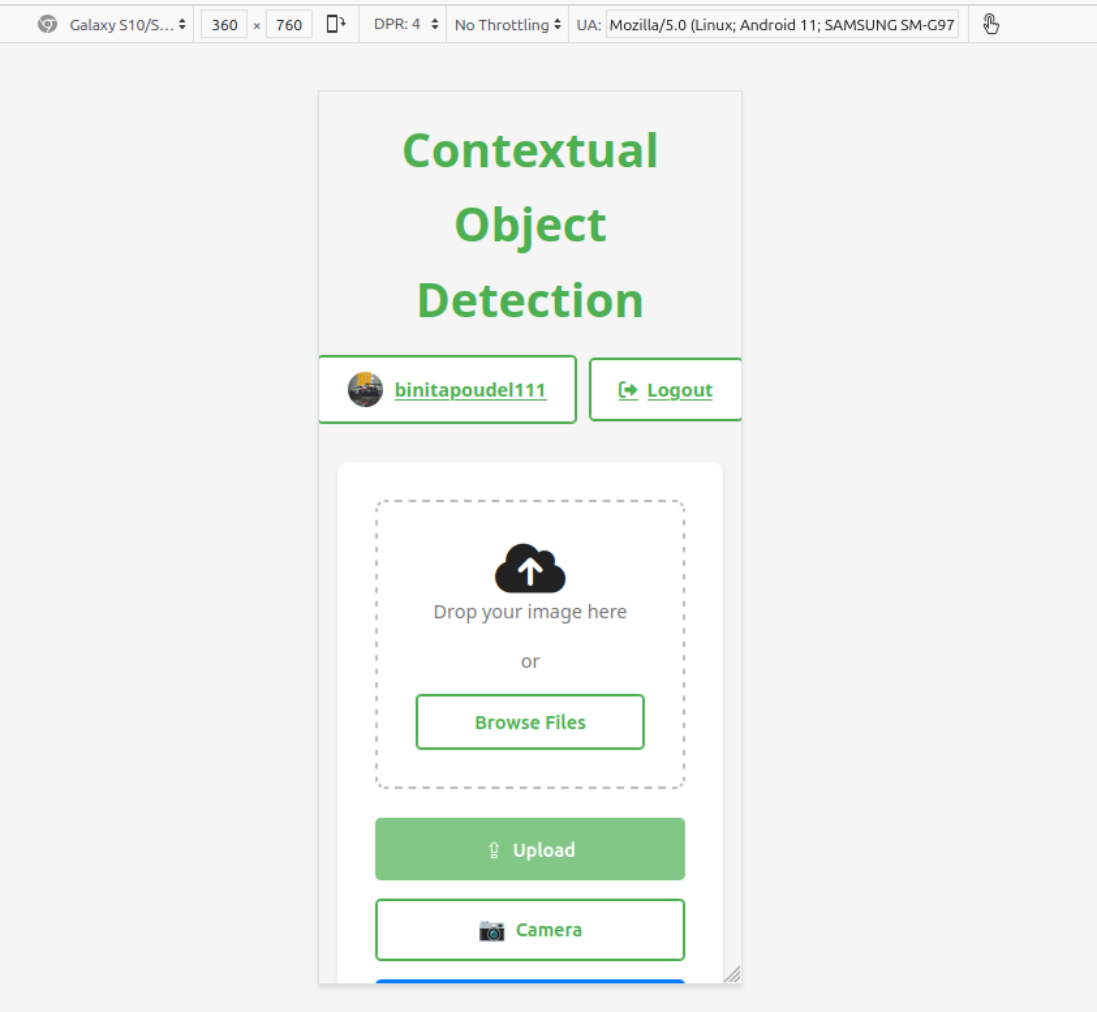


Figure 131: Main page in tablet configuration

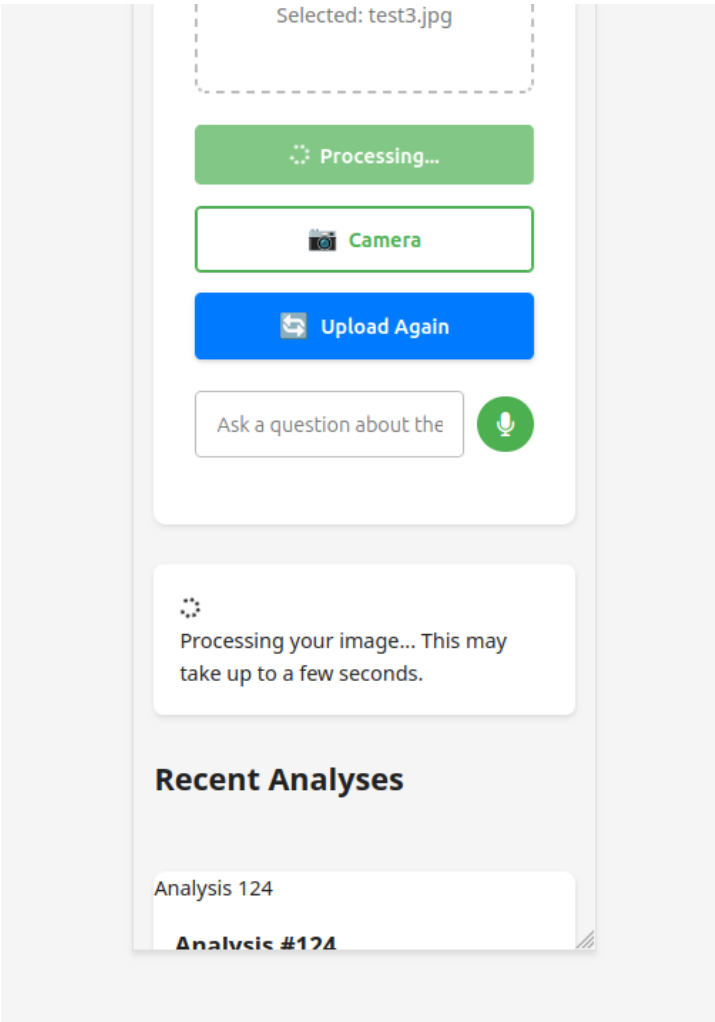


Figure 132: Context generation in mobile configuration

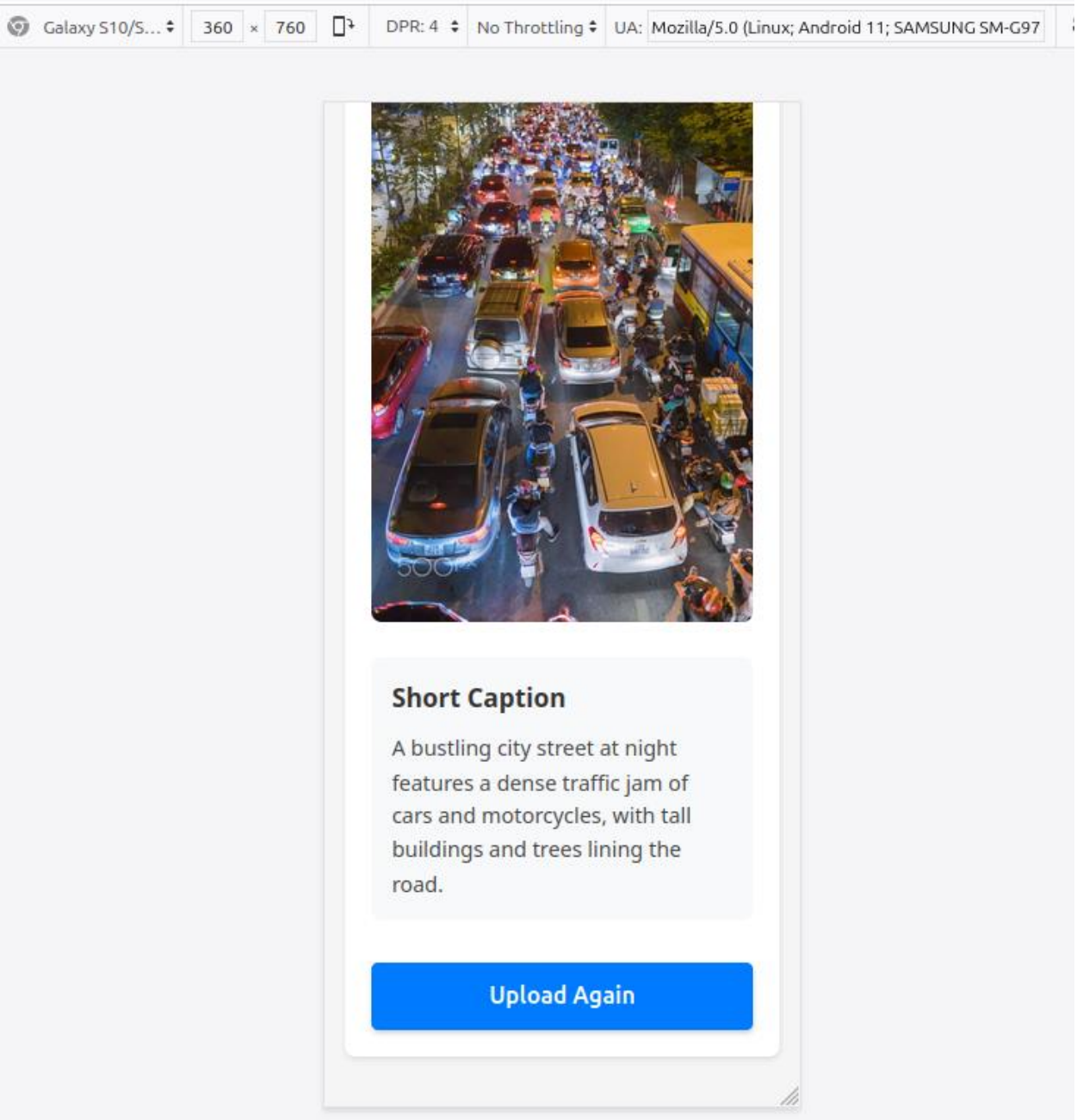


Figure 133: Caption generated in tablet configuration

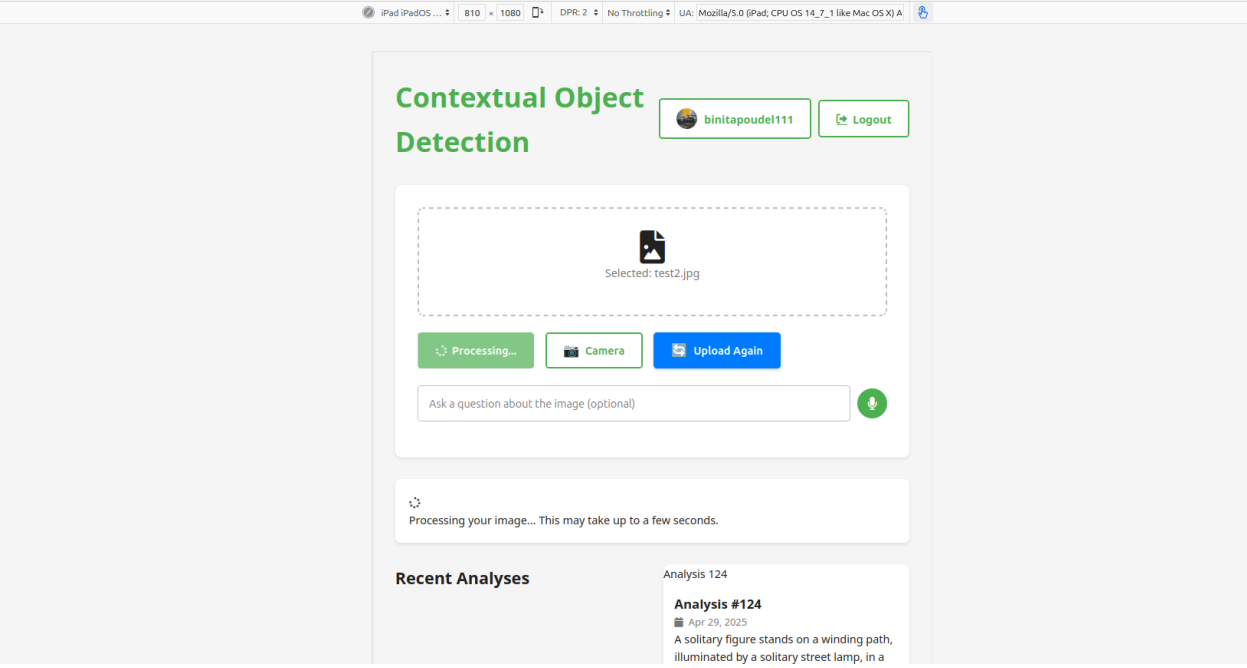


Figure 134: Uploaing image in tablet configuration

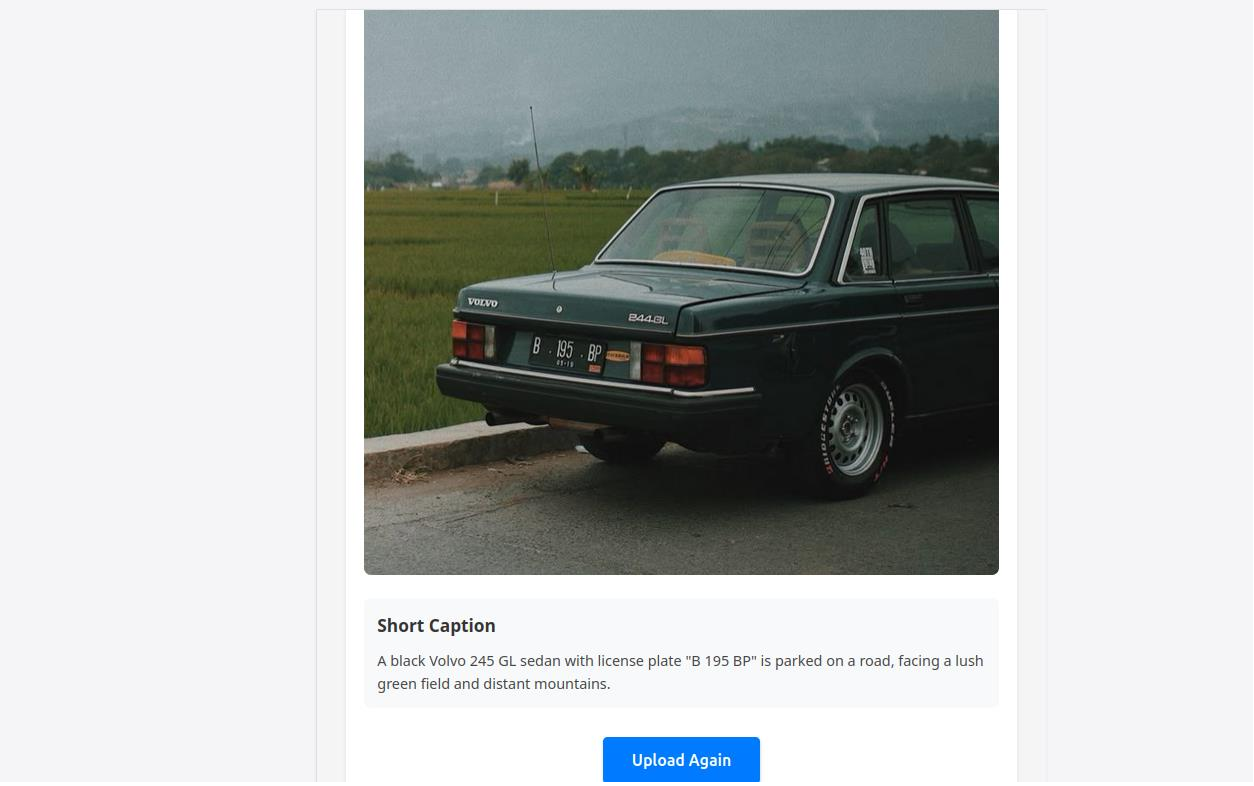


Figure 135: Generated context in tablet configuration

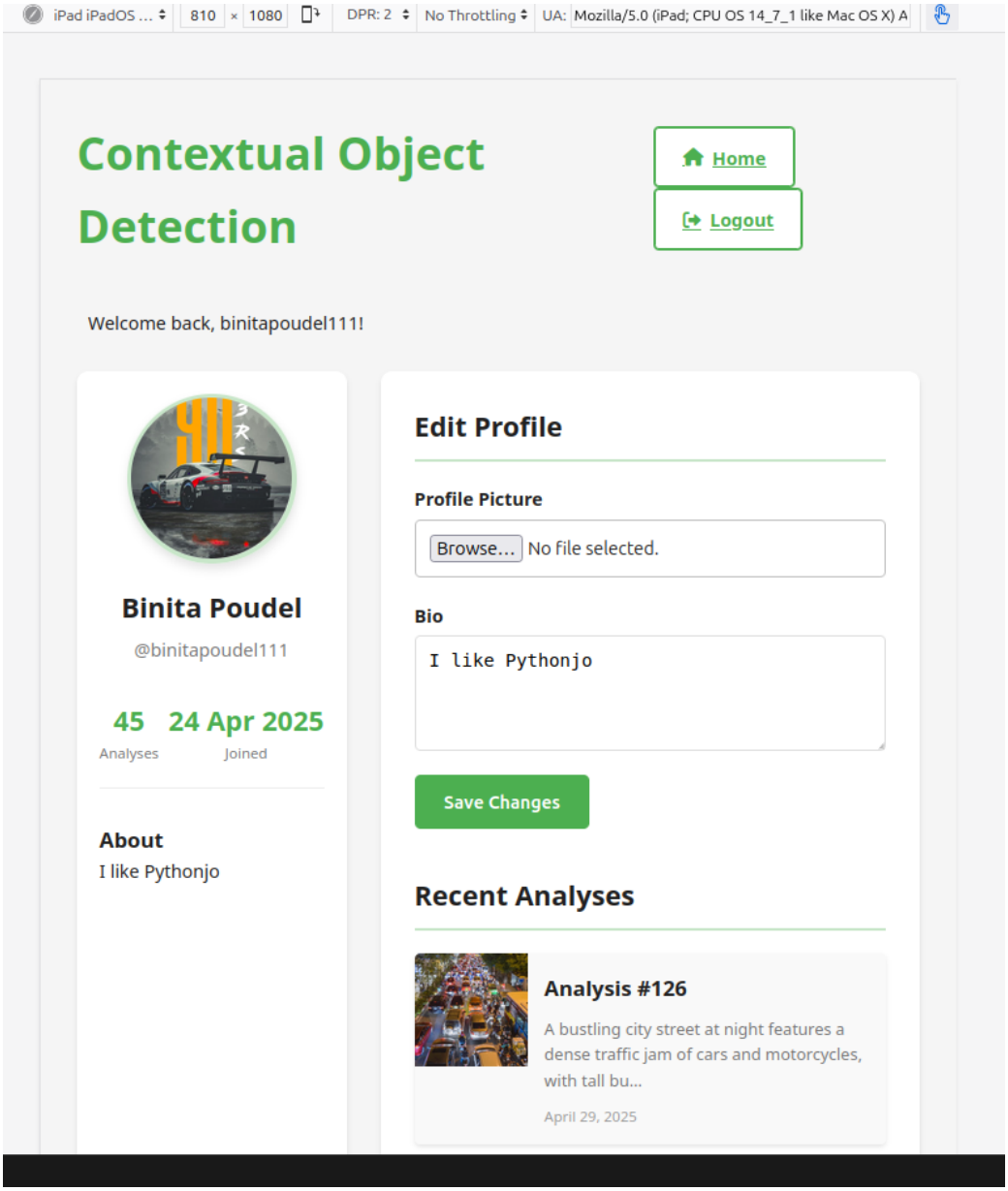


Figure 136:Users profile in tablet configuration

4.3.5. Test 5: Image processing Pipeline and different image file format

Objectives	To verify the that the system accepts wide range of image format and gracefully deals with the unsupported format.
Action	I. Logged in to the system II. Uploaded images with extensions: jpeg, jpg, png, bmp, tiff.
Expected Result	The system should process the images with supported extension and gracefully dealt with unsupported format.
Actual Result	The
Conclusion	Test passed.

Table 34: SysTest 5: Image processing pipeline and differetn file format

Screenshots

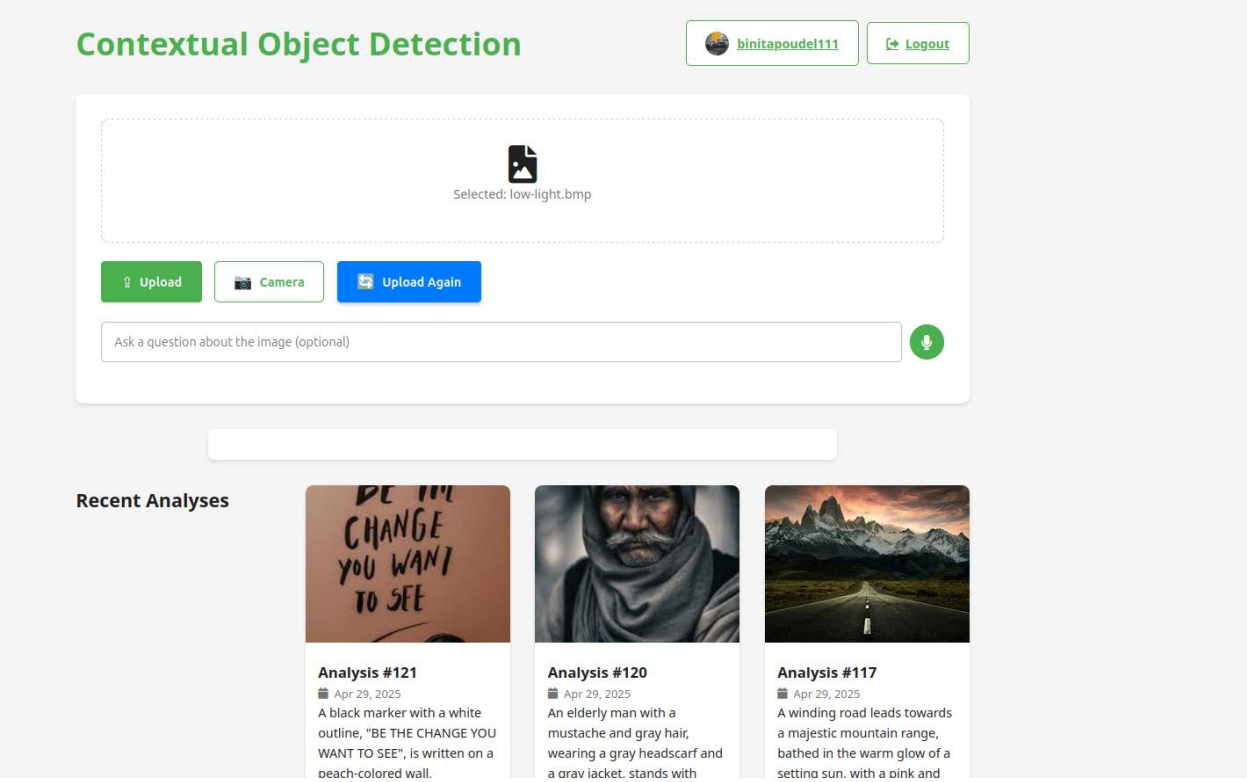


Figure 137: Uploading bmp image

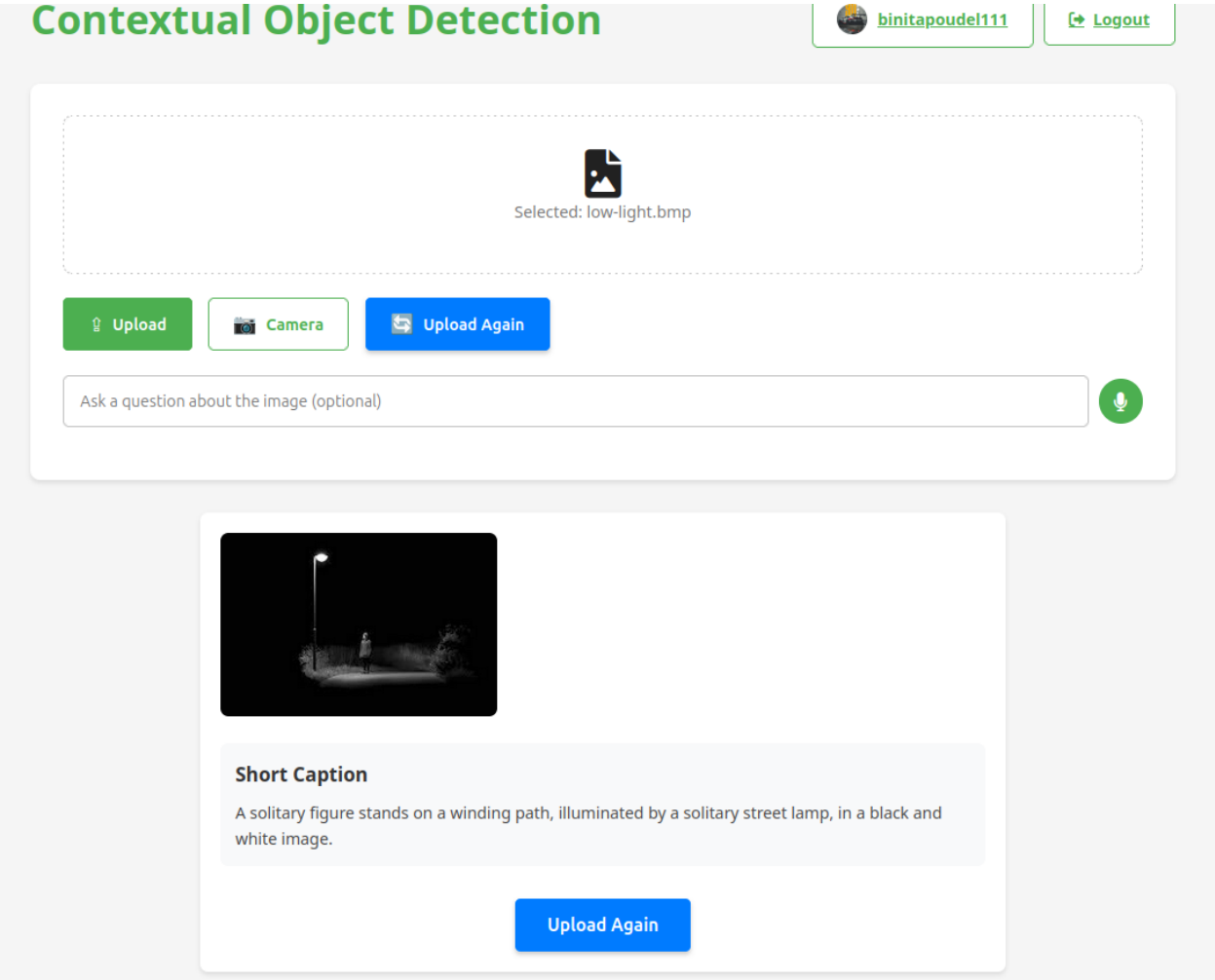


Figure 138: Context for bmp image

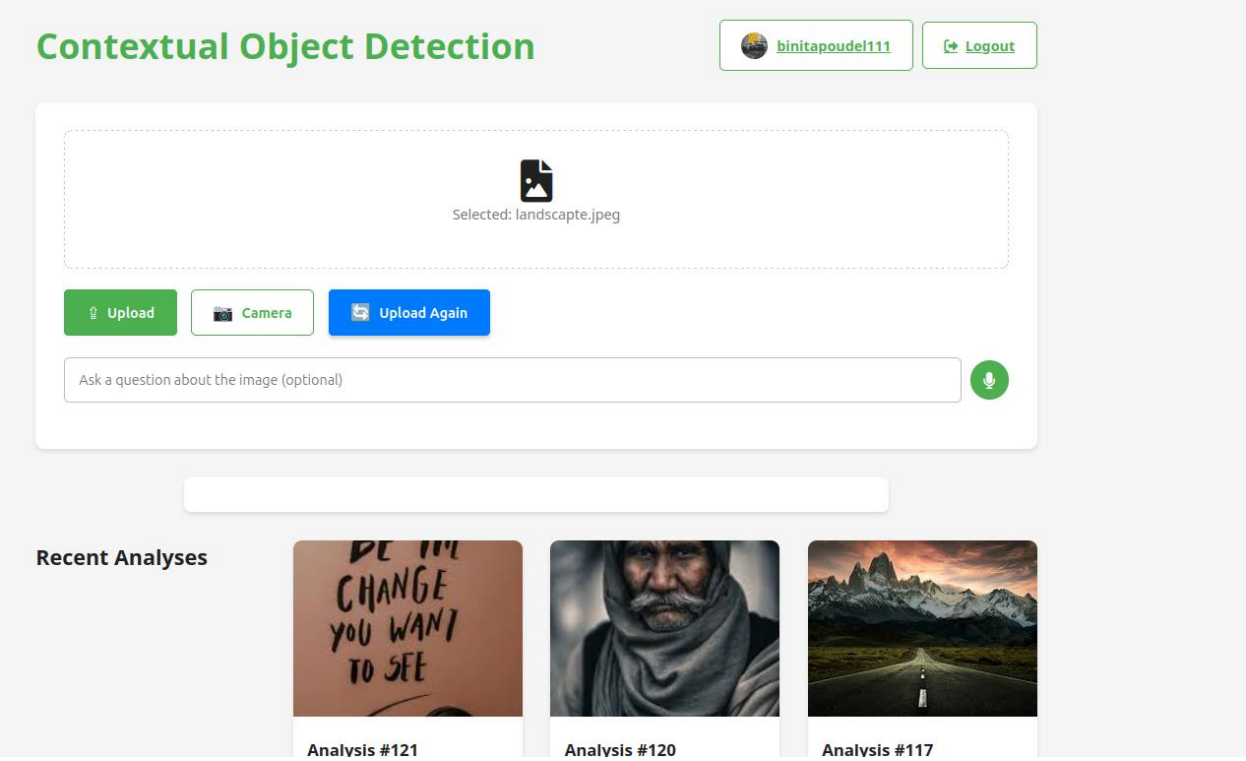


Figure 139: Uploading jpeg image

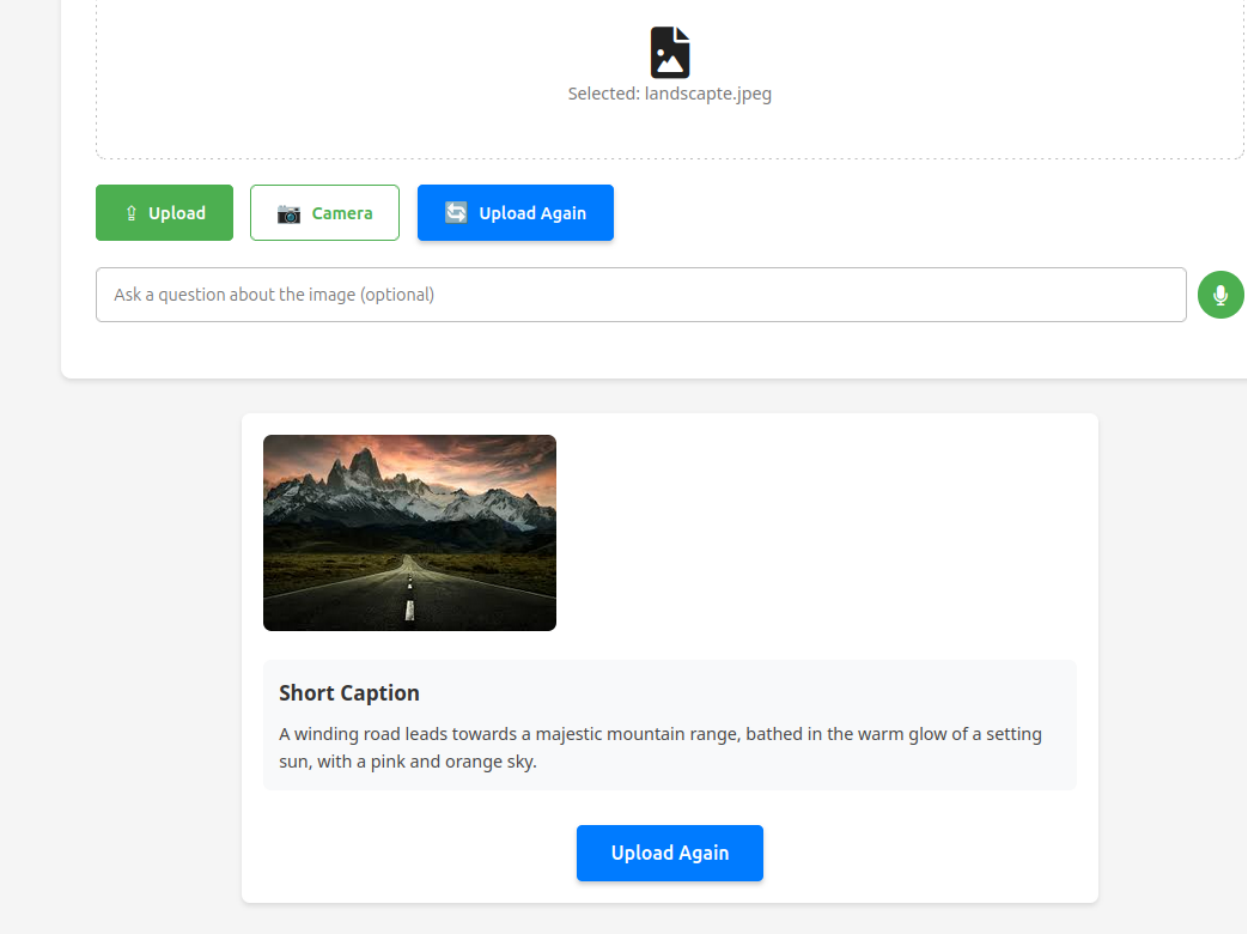


Figure 140: Context for jpeg image

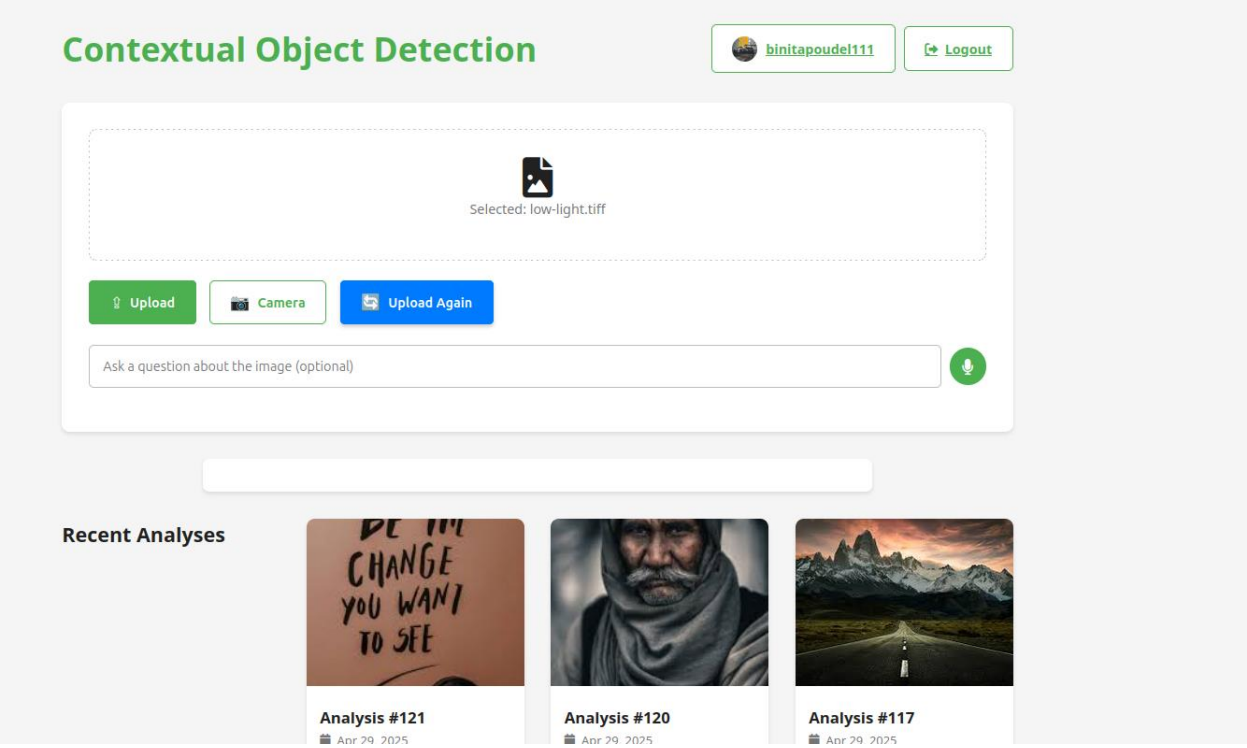


Figure 141: Tiff image upload

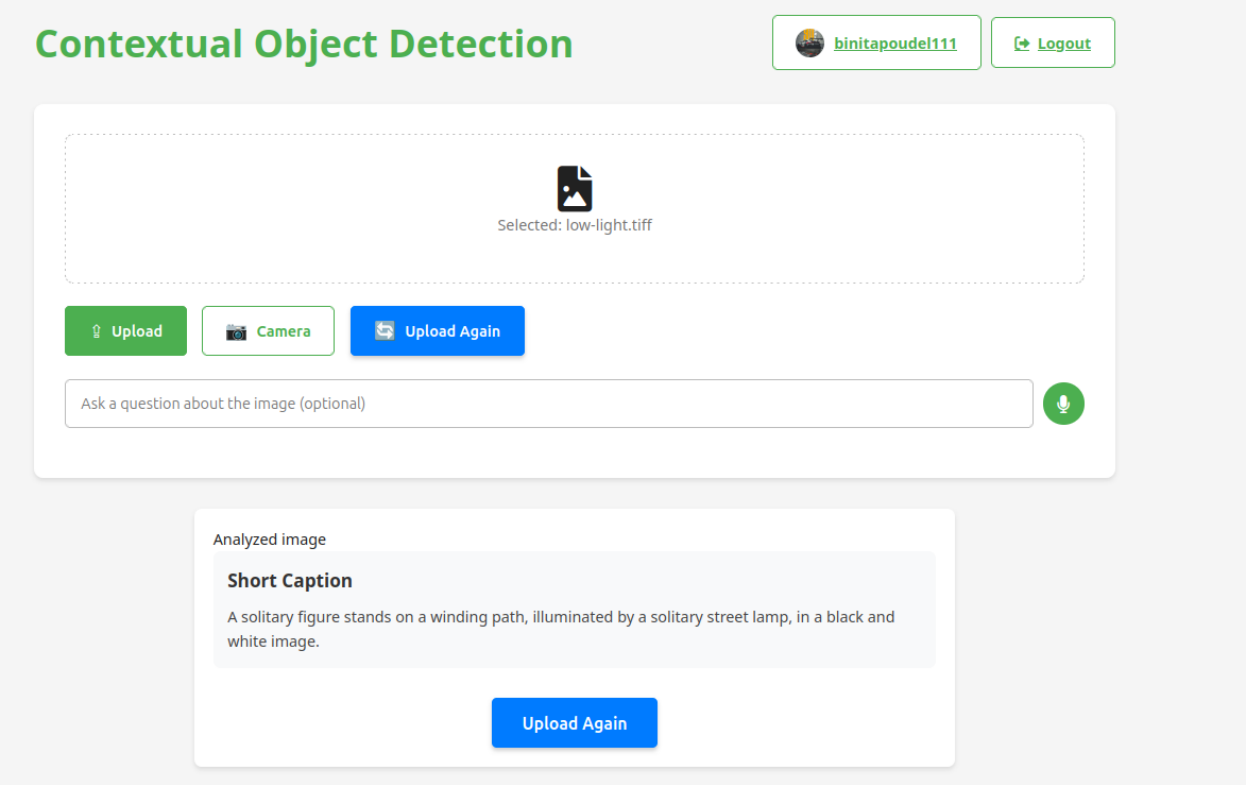


Figure 142: Context for tiff image

4.3.6. Test 6: Running System on Linux Operating System

Objectives	To run Contextual Object Detection in Linux Operating system.
Action	Run Django and Redis Server.
Expected Result	Contextual Object detection should run in Linux platform, every functionality should work within the application.
Actual Result	The system ran in Linux Mint without any issues and every feature was functional.
Conclusion	Test passed

Table 35: SysTest 6: Running System on Linux System

Screenshots

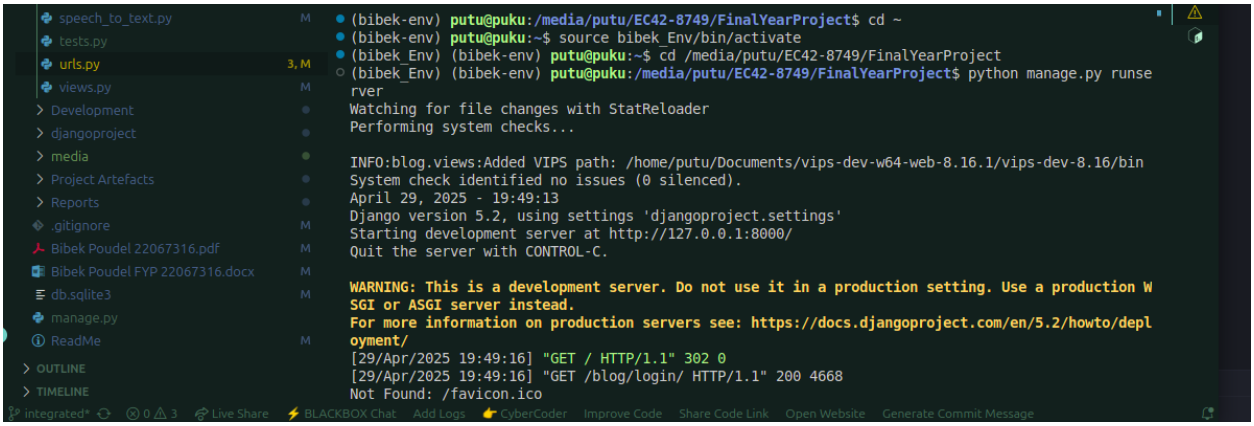


Figure 143: Running system on linux machine

4.3.7. Test 7: To test the context generated from the model on benchmark dataset as ground truth

Objectives	To test the context generated from the model on benchmark dataset as ground truth
Action	I. Use the images and ground truth caption from the benchmark dataset from Hugging Face II. Generated the context for the same image using our model
Expected Result	The model should generate very detailed and accurate captions for the medical images provided with respect to the ground truth caption.
Actual Result	The model generated very detailed and accurate captions for the medical images provided with respect to the ground truth caption.
Conclusion	Test passed.

Table 36: SysTest 7: To test the context generation from the model on benchmark dataset as ground truth

Screenshots

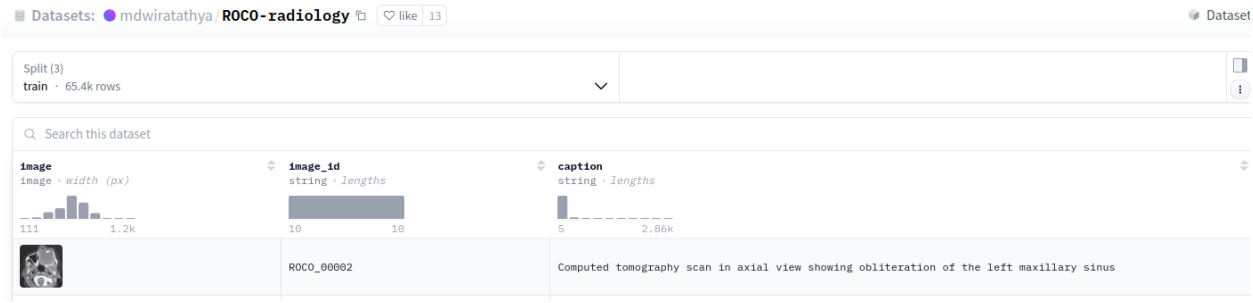


Figure 144: First benchmark image-caption pair

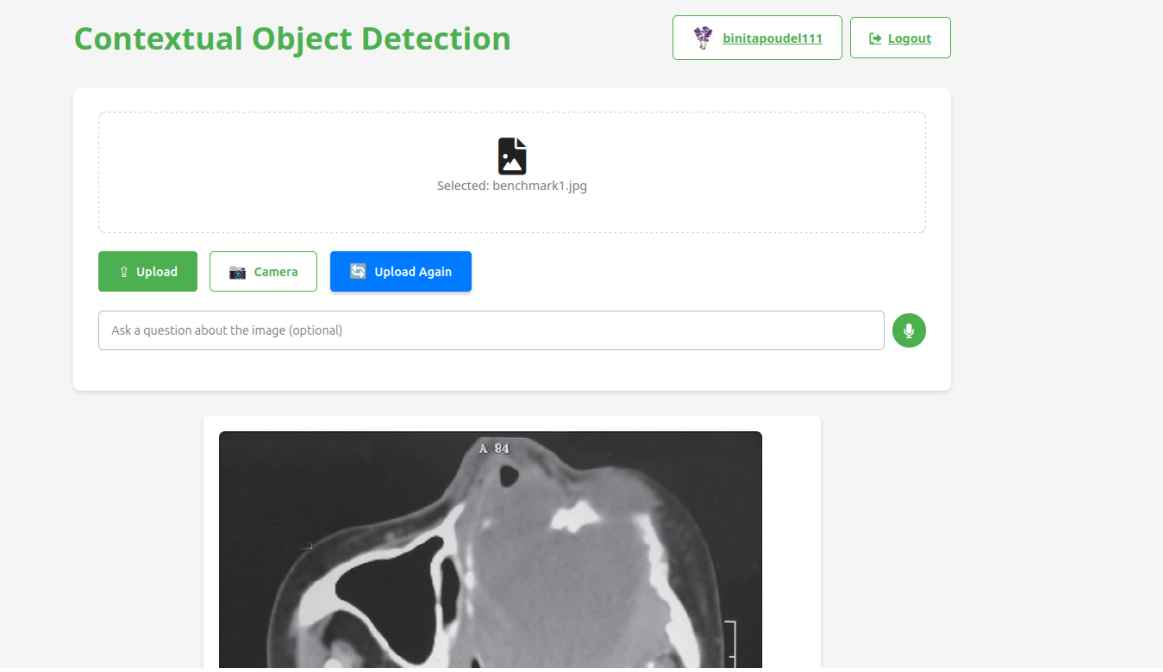


Figure 145: Context generation for benchmark 1

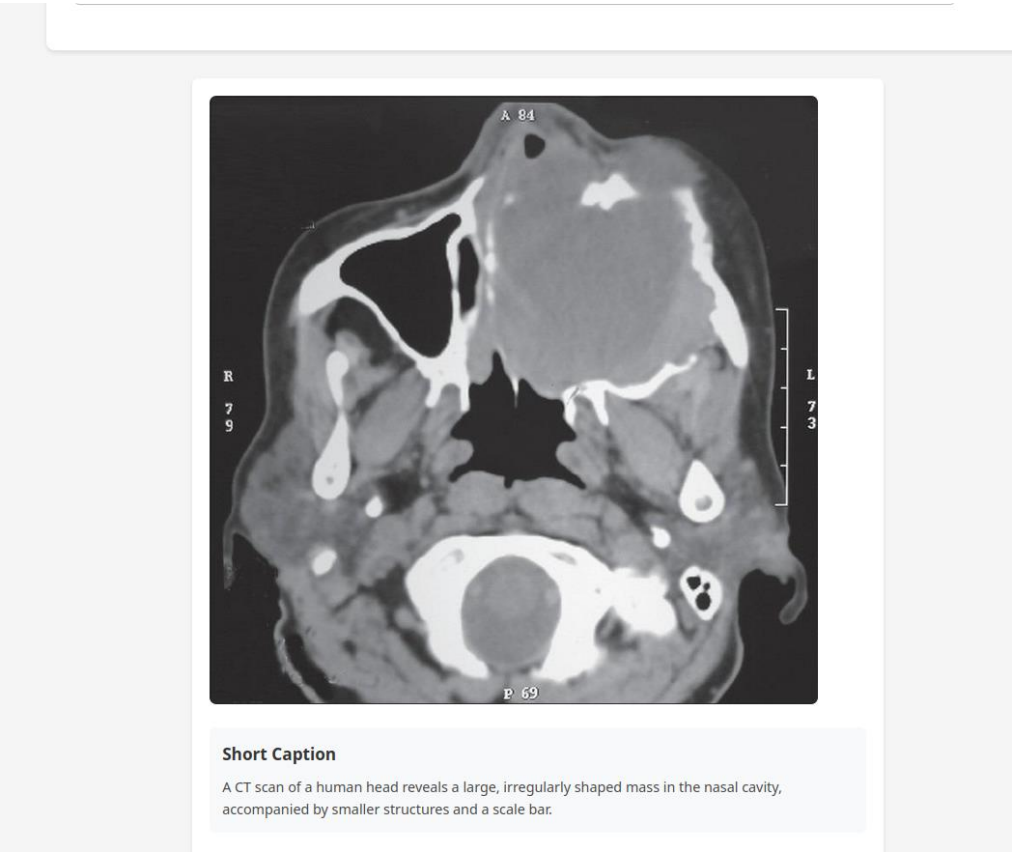


Figure 146: Context generated for benchmark 1

Figure 147: Ground truth image-caption pair for benchmark 2

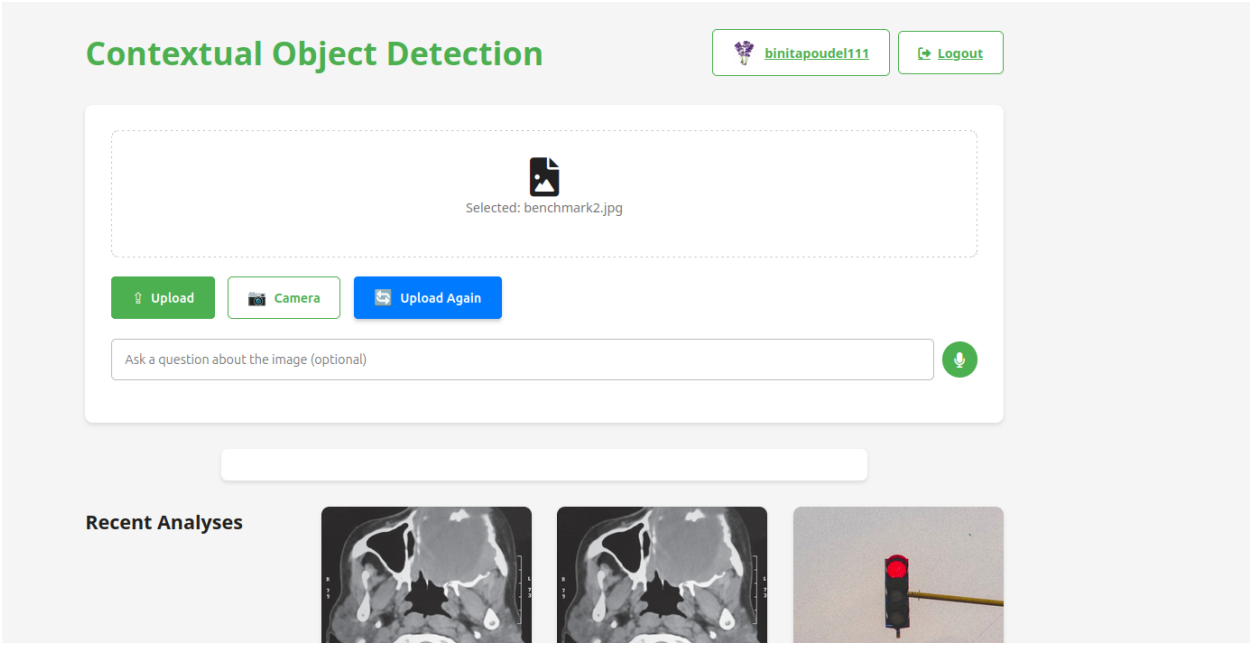


Figure 148: Uploading benchmark 2 image

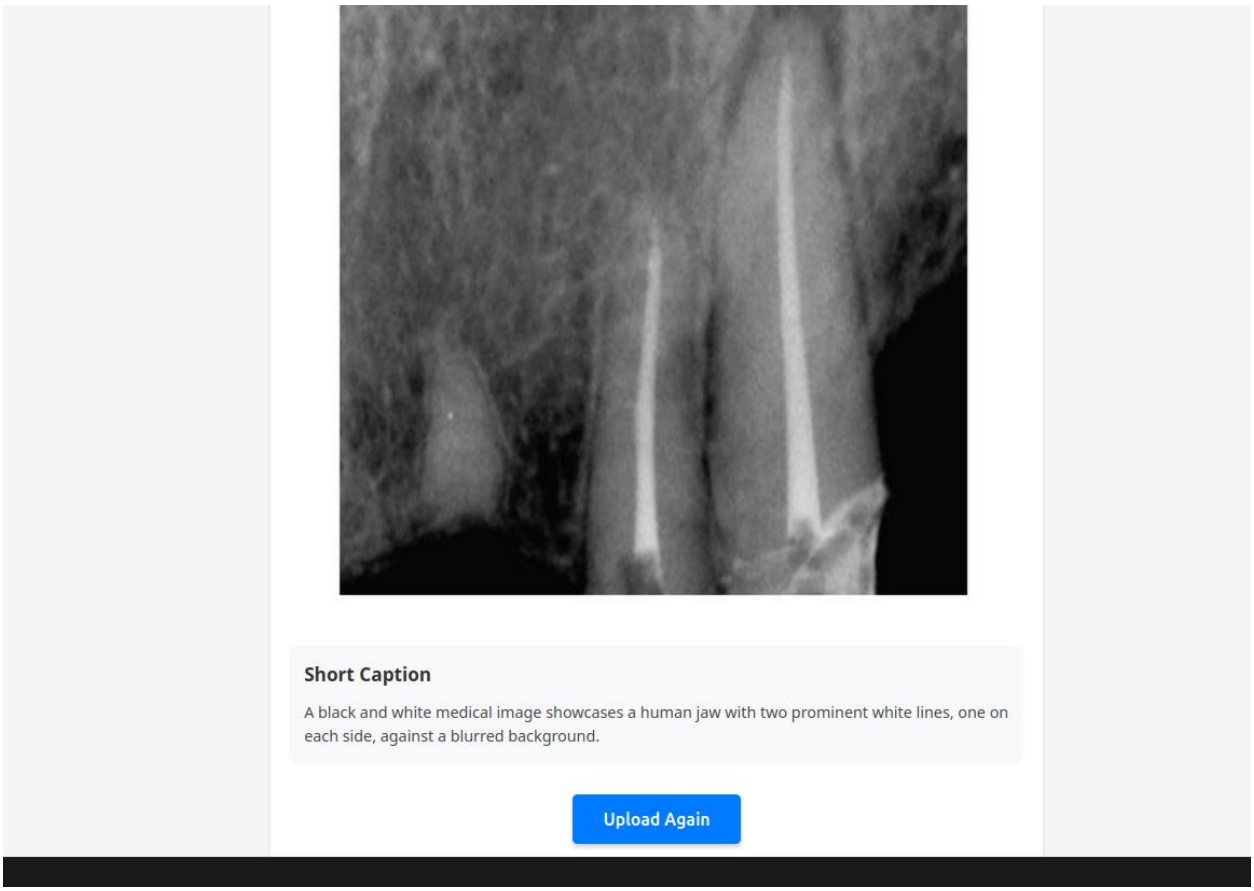


Figure 149: Generated context for benchmark 2

	ROCO_00010	Post orthodontic treatment. Root canal therapy done with maxillary incisors

Figure 150: Third ground truth and image

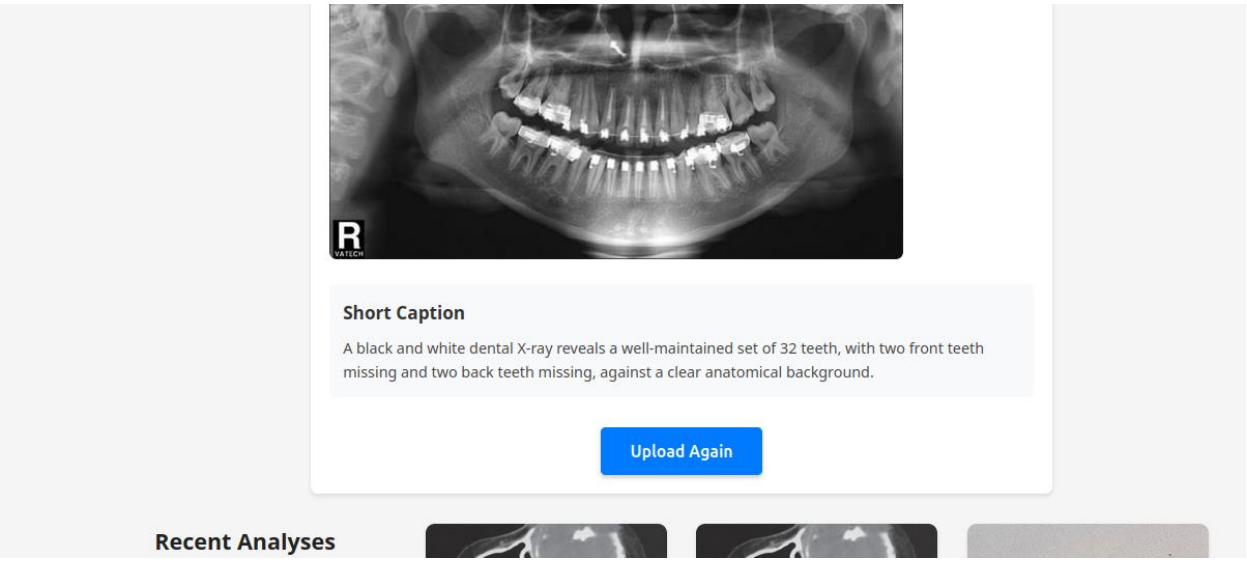


Figure 151: Generated context for third benchmark image

4.3.8. Test 8: To test admin's ability to view admin dashboard and access exclusive features.

Objectives	To test admin's ability to view admin dashboard and access exclusive features.
Action	Login to admin panel and view dashboard.
Expected Result	Admin should be able to list out and perform CRUD operations on user's data and analyses images and their context along with the viewing the insights of users, images analyzed and queries processed by the web-application.
Actual Result	The admin was able to perform the CRUD operations on user's data analyzed image's data along with viewing the summary of detections and users.
Conclusion	Test passed.

Table 37:SysTest 8: To test admin's ability to view admin dashboard and access exclusive features

Screenshots

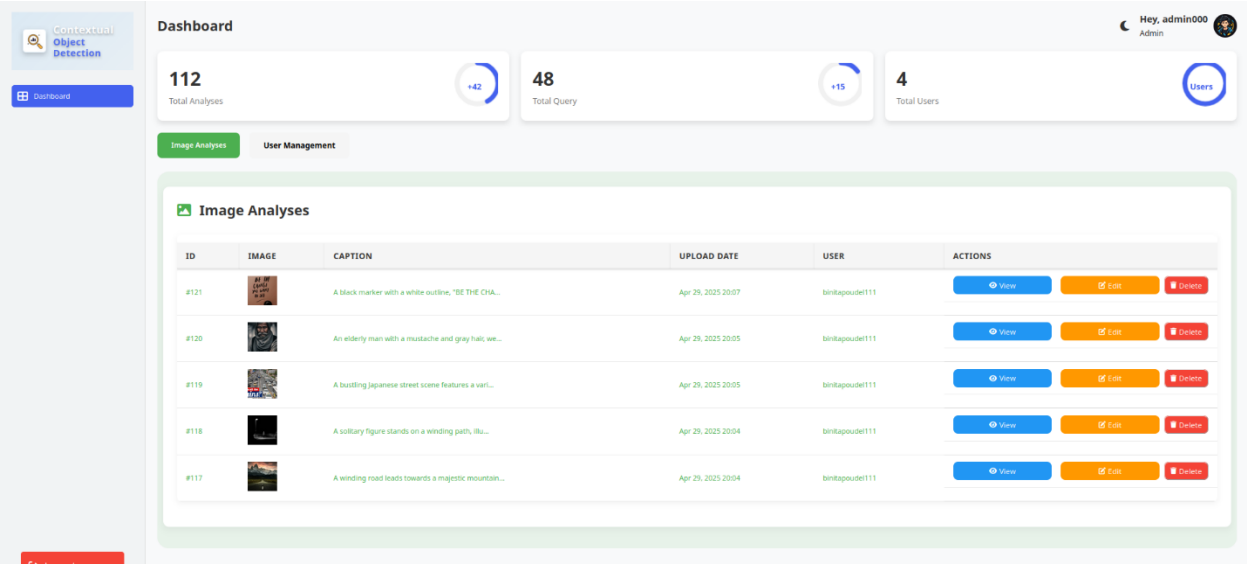


Figure 152:Admin dashboard

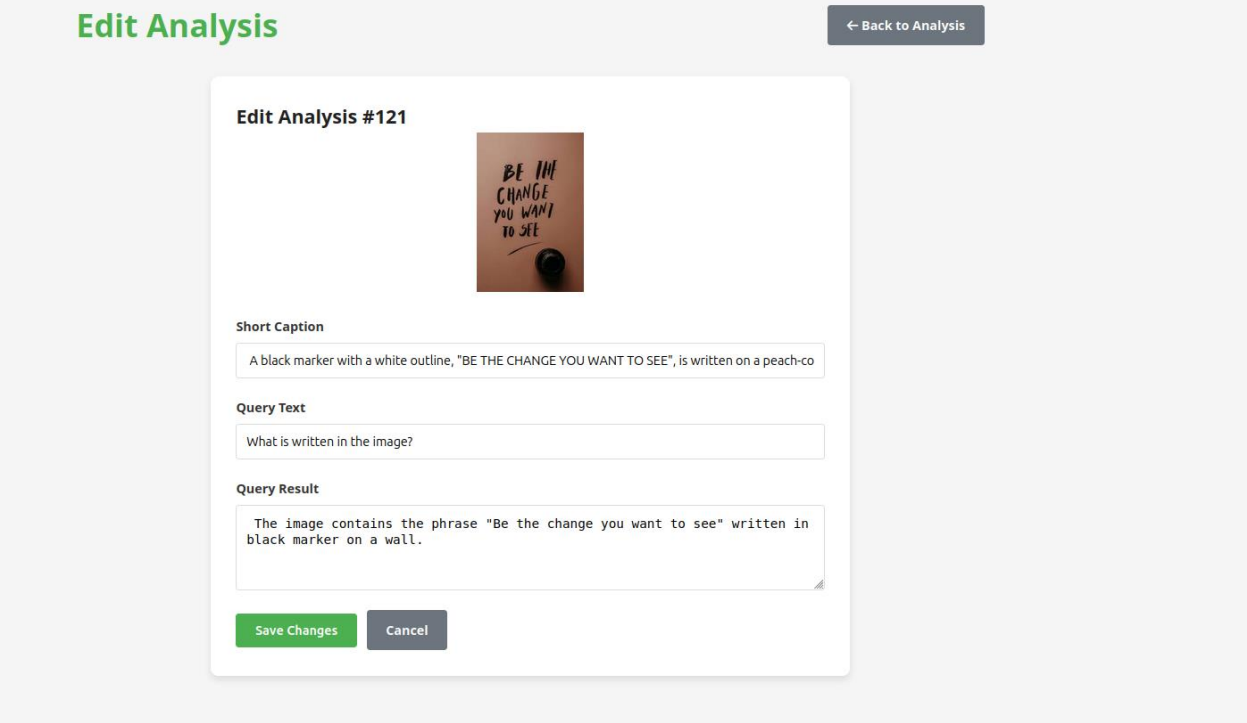


Figure 153: Admin editing image analysis

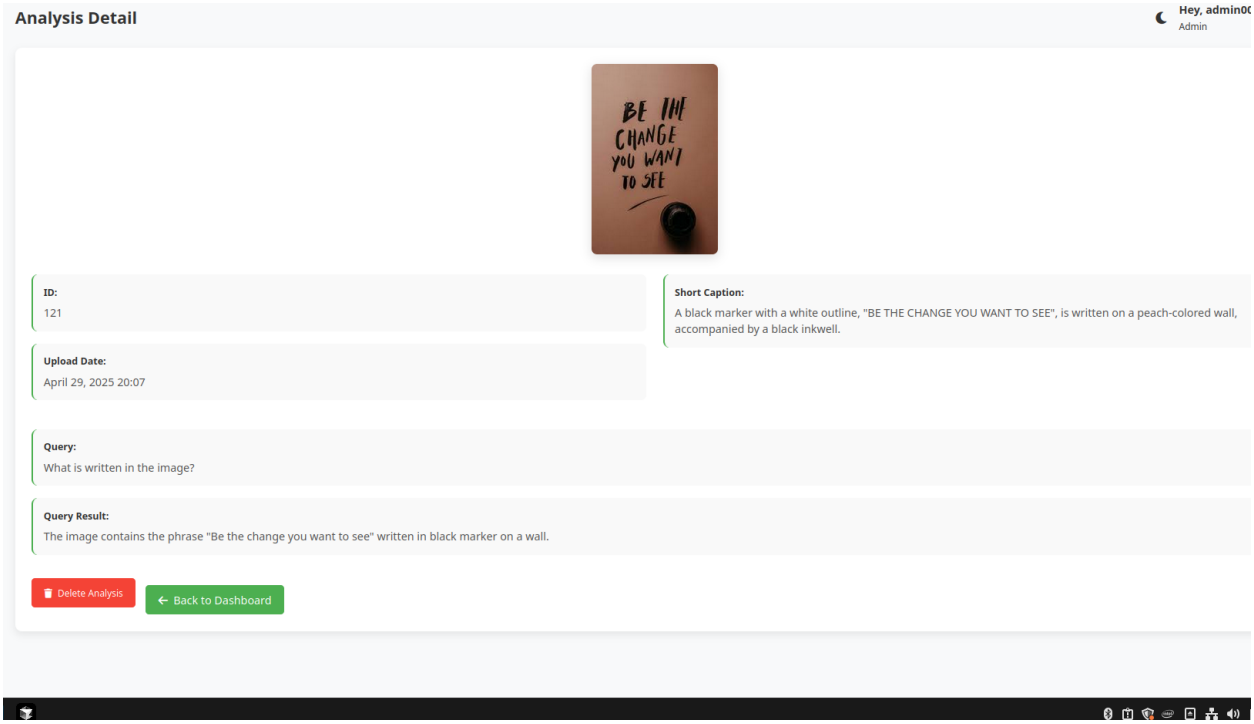


Figure 154: Admin viewing image analysis

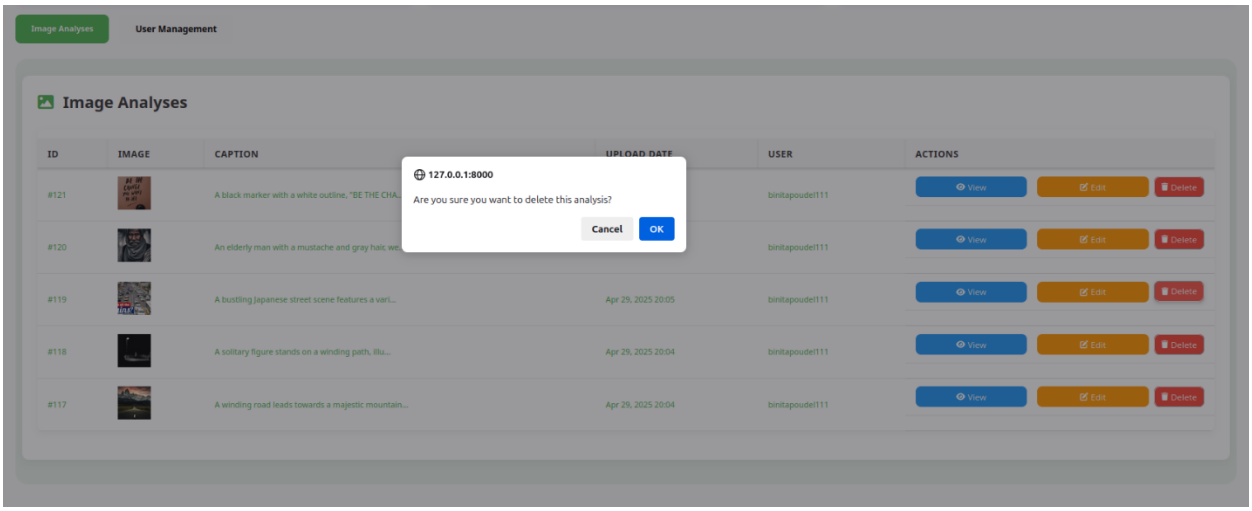


Figure 155:Admin deleting the user

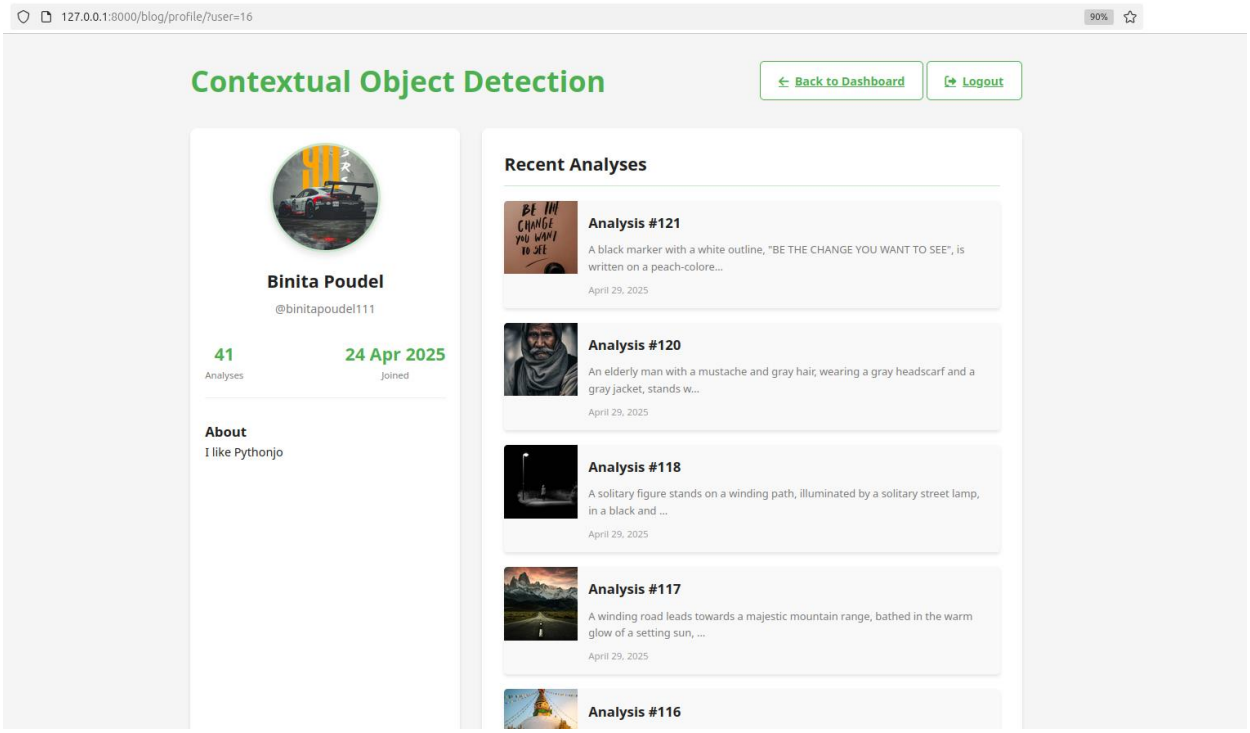


Figure 156:Admin viewing user details

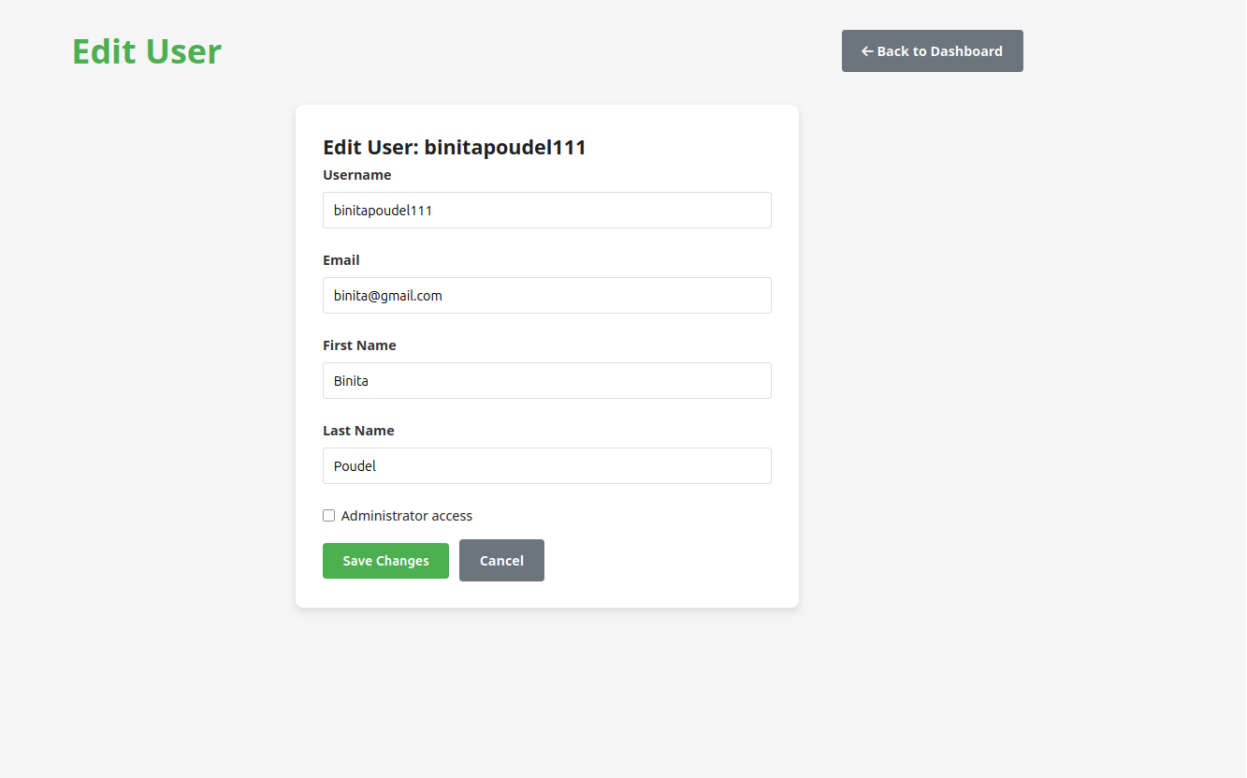


Figure 157:Admin edits user's details

4.4. Testing Phase Summary

This test plan covers testing the Contextual Object Detection system, with particular attention to the Sprint 3 features and end-to-end capabilities. Key features to be tested/screened are user authentication (login, registration, profile update) and context creation through uploading/capturing images and processing in real-time and through speech-to-text integrated processing. Tests emphasize contextual modelling, consistent with recent advances in multimodal large language models for human-AI interactive understanding of perceptible objects. Critical use cases like handling of invalid/corrupt files, visual query responses, and the ability to delete were rigorously tested for robustness. Cross-feature interactions and Redis-driven background tasks proved the smooth integration of Django-PyTorch while Linux compatibility testing ensured deployment scalability, which is an absolute necessity for deep learning workflows on top of such heavy-computational frameworks.

Role-based functionality in the context of secure data governance was ensured using a hack into the admin dashboard, i.e., rights and privileges look up against users. Performance evaluation was

based on context generation speed and accuracy on benchmark data, thus following the DART-like procedure from experimental evaluation. Redundant checks (e.g., profile picture visibility) + edge cases (broken images) contributed to the robustness, reminiscent of approaches where graph contextual reasoning benefits outlier identification in complex scenes. Cross-device UX testing made the experience responsive across devices, also a good fit with agile practices that constantly validate AI-based pipelines.

4.5. Critical Analysis

Results for the first round of testing Contextual Object Detection showed interests and limitations in adding YOLOv8 to multimodal systems. These initial tests focused on the detection accuracy and inference rate of YOLOv8 since LLaMA, Phi, and Gemma demonstrated general language capabilities, albeit at poor domain-specific contextual reasoning when compared with vision language models (VLMS), like LLaVA and Moondream2. Unexpectedly, the compact design of Moondream2 could effectively bridge the trade-offs between performance and efficiency, which agreed with the design philosophy of GPU memory optimization. But there was a latency bottleneck in real-time processing for speech-to-text integration, which prevented smooth user interactional challenge shared by multimodal pipeline evaluations. Combining Django and PyTorch presented technical difficulties in how to handle Redis-driven background tasks for context generation, which at the beginning would last 20-40 minutes per image due to CPU-bounded computation. This is reduced to 4–10 seconds with GPU acceleration, emphasizing the criticality of hardware-software co-optimization.

Web app security, like auth flows and input validation, was heavily enforced by Django middleware, however, the edge cases (a corrupted file upload, for instance) could reveal a small security hole in error handling and the like but mostly requiring “bool” protection in object pre-processing. These advances mark an incremental step forward in the state-of-the-art of AI systems, where architecture and infrastructure optimization change whether a model is usable for real-world applications.

5. Chapter 5: Conclusion (Legal, Social & Ethical Issues)

5.1. Legal Issues

5.1.1. Intellectual Property and Copyright Considerations

The Contextual Object Detection page must also comply with intellectual property law for the user-uploaded images and analysis of copyrighted visual content. The website is to store all these pictures that can reasonably be assumed to contain copyrighted material. It obeys in compliance with the law, the site should have clear terms of service that claim ownership of the uploaded content, establish its criteria for acceptable use and contain notice and take-down procedures for copyright holders. What's more, the platform also needs to think about content filtering so that it's not feeding anything potentially infringing into the system and giving copious credit to any third-party AI models or libraries in use within the system itself, such as the Moondream2 model.

5.1.2. Data Protection and Privacy Regulations

The Contextual Object Detection system captures and processes large amounts of user content, personal information, images, and results of analysis performed on the input data. There is a requirement regarding data protection: secure your data as much as possible (encryption, managing databases). The platform's data retention policy, especially for stored images and analysis results, must be clear and explicit. Users may also need to be notified about which data will be collected, how it will be used, and its sharing restrictions, with explicit privacy notices. System design, which involves user profiling by collecting personal data, should care for the principles of data minimization, proper consent handling in the system.

5.1.3. AI and Automated Decision-Making Regulations

In offering a computer vision platform that performs automated image analysis, as a company, it must walk the line of new laws about automated decision-making systems. Classical CV object detection and captioning models also need to comply with the transparency of algorithms. The system prompts must make it noticeable that the AI is generating product or analysis, particularly for image captioning and query response features. Documentation and version-controlled performance metrics for the system's machine learning models, like the Moondream2 model, must be retained to show due diligence. End users need to be provided with meaningful information on the process of analysis, and warnings on the efficacy of the result.

5.1.4. Digital Accessibility Compliance

The object detection by context should comply with the accessibility standards to make it accessible to people with disabilities. The support for different input modalities, such as speech-to-text and camera, is indicative of an attempt towards accessibility guideline compliance. Moreover, the platform should provide analysis output in formats, such as providing text alternatives for visual media. ARIA attributes are necessary for proper desktop and mobile UI elements as well as keyboard navigation support. Continued accessibility testing efforts should be conducted to identify and address barriers to access, especially regarding the modal interfaces and interactive elements of the service.

5.2. Social Issues

5.2.1. Digital Divide and Technology Access

The implementation of the Contextual Object Detection framework in Nepal faces the digital divide challenges that affect the adoption of modern computer vision technologies. The online system needs a reliable network to log into, which can limit its applicability in areas with poor infrastructure, such as most rural areas. The greater resource efficiency of the platform, including resource-efficient use of CPU and the ability to store models on-device, is also promising in terms of addressing the hardware bottleneck. To reduce barriers to access, the platform can improve offline experience, support progressive loading on slow networks, and optimize the interface for low-end smartphones. Training and tutorials in local languages could close the knowledge gap for those unfamiliar with AI-supported image analysis technology.

5.2.2. Cultural Context and Representation

AI models, through processing and interpreting visual data, are naturally biased in certain objects' identification from these different cultural contexts. Object detection and image captioning on the platform could issue culturally biased results if the models are not trained with data that captures representative features. The platform must also take steps to gather and annotate images globally, beyond the limited number acquired from a single hospital based on one racial group and validate model performance against cultural diversity. Furthermore, local Nepali image analysis algorithms need to be tested in local Nepali visual conditions for culturally constrained object, environment and situation recognition. Further, iterative development processes ought to involve

feedback from a variety of user communities to make cultural sensibilities feasible for object detection and description generation.

5.2.3. User Trust and Technology Perception

Public acceptance and attitude towards artificial intelligence technologies as object detection and image analysis systems do also play a role in the way they are used and adopted. The transparency features, through the provision of leading indicators for data processing and of detailed presentations of results, are also crucial for trust-building of users in the platform. But concerns about the accuracy of AI systems and privacy issues could challenge adoption. For building and keeping trust, the platform should be transparent about what it can and cannot do, including in terms of detection accuracy. The inclusion of features that support viewing results at all scales and analysis history aids users in understanding and validating system results. Providing tutorial resources describing the operation of image-analysis technology would help to remove the mystique associated with such technology and permit users to make informed decisions regarding its use.

5.3. Ethical Issues

5.3.1. Data Privacy and User Confidentiality

The platform is an integration framework that interfaces with potentially privacy-sensitive user-generated visual content, causing major privacy and confidentiality-related ethical issues. Storing the images and image analysis results in user histories in the system design is a moral responsibility for the system to refrain from unauthorized use and access to such information. It is the responsibility of the platform to enforce multi-levels of privacy protection that include secure modes of authentication, data-encrypting techniques, as well as fine-grained access controls. There should be well-defined data life cycle policies governing if and for how long images and results are stored. Users ought to have full control over data, including the ability to delete past analyses. Design thinking (from the very frontend to the most backend) shall come with the default that privacy is a priority and has everything to do with it.

5.3.2. AI Accuracy and Misidentification Risks

The adoption of image and object-detection technologies raises ethical concerns regarding their accuracy and the risks associated with misidentifications. The object detection algorithms the platform uses are prone to false positives and false negatives, which means the impressions can misinform users who depend on these estimates. Its design incorporates confidence scores on detected objects, which reflect a willingness to be transparent about uncertainty. To address accuracy-related issues, the platform needs to conduct a rolling assessment and enhancement of model performance for various image formats. Clear caveats on the limitations of accuracy ought to accompany these results. The former failure modes may be monitored and corrected, for example, by periodic QA tests. The platform should also enable users to report faults in the analysis to pursue continuous feedback/honing.

5.3.3. Accessibility and Inclusive Design

The COD (Contextual Object Detection) platform must address ethical issues and guarantee equal access and use of it for different user groups. Intersect multiple input modalities, for example, speech-to-text or camera, as a demonstration of inclusive design principles. But the platform still needs to overcome accessibility barriers for people with disabilities, people who do not have digital literacy, or people who speak fewer of the major languages. The user interface should adhere to generally recognized design conventions, should be screen reader compatible and should display text alternatives. Functions with voice input should be implemented with alternative voice patterns and accents. Furthermore, the system needs to be evaluated with users of different ability levels, age groups, and computer expertise to identify and eliminate any usability issues. By promoting inclusive design practice, the platform can live up to its ethical responsibility of providing equitable service to its diverse target users.

5.4. Advantages

5.4.1. Resource-Efficient Visual Intelligence

The Contextual Object Detection model is very effective in providing intelligent computer vision capabilities using very low computational power and memory. Unlike typical generic AI systems, which require a massive GPU setup to work, the best part of the system has been designed to be able to run on standard CPU hardware. A multimodal architecture with local model storage and

CPU-optimized inference reduces processing time and resource usage, providing high-level visual inspection on typical computers to users. Such an optimal performance is very relevant to the scenarios of Nepal, where limited and costly access to high-performance computing machinery by passers and organizations remains a challenge.

5.4.2. Comprehensive User Experience

The solution provides a full end-to-end experience beyond just basic object detection to elevate the entire user experience. The user account and profile management system, in combination with the analytic history, supports a custom environment where users may track their analyses and continue work from previous analyses. The addition of mode views of old analysis fosters the possibility of users being able to see and refer to old results, and thereby develop a longer-lasting understanding of visual content. This holistic technique allows object detection to be freed from a siloed technical capability to become an entire application, integrated into reality.

5.4.3. Multi-Modal Interaction

The system can accommodate multi-modal inputs for improved user convenience and accessibility. In addition, interaction capabilities include upload from a camera, direct upload of images, and speech-to-text Entry, making the service versatile for use in various deployment scenarios and personal preferences. Audio commands via speech-to-text allow for hands-free control in situations where it would otherwise be impossible to provide any input. The camera function also enables visual analysis of scenes, which obviates separate processes for image capturing and uploading. The multi-modal strategy covers all users' different requirements and diverse application scenarios, demonstrating a user-experience-oriented design guideline.

5.4.4. Open-Source Adaptability

Open-source context the open-source character of Contextual Object Detection provides great value in terms of customizability, expandability, and collaborative improvement. The system architecture allows for interchange of models to satisfy various detection needs, or to trade and balance accuracy and processing speed. Organizations can also modify the codebase to work with their existing systems or add custom-specific features to serve domains. This flexibility is interesting in the Nepali context, where there may be a need to localize and to adapt for cultural

or linguistic purposes. And the open-source model encourages local knowledge sharing and group-based innovation in the tech industry.

5.5. Limitations

5.5.1. Detection Accuracy Constraints

Although the Contextual Object Detection system enjoys strong detection performance, it also has some limitations on the detection accuracy in challenging conditions. It is not effective in the presence of low light, partial occlusion or highly overlapping parts. Its adoption of pre-trained models causes it to inherit any biases and weaknesses of the original training data, which may similarly impair the detection of objects, scenes, and contexts which have received low exposure in pre-trained model training. The limited accuracy of measurements is the motivation to educate the users (limitations of the system) and to be very careful with proper applications.

5.5.2. Processing Time and Scalability

The system works well for single users, however, it becomes operationally intensive as the system scale reaches or exceeds capacity and desired timescales. In general, conventional computing (CPU-only) will suffer from longer processing time for big images combined with complex scenes. The job queue functionality using Redis is efficient in handling concurrent processes, but it hits its performance cap when the system activity increases to reach full capacity. The system is not efficient in rapid-deployable or very high processing volume situations.

5.5.3. Technical Deployment Barriers

The Contextual Object Detection system requires specific technical knowledge, ch may be an obstacle for a prospective user or a company in using the system effectively. The level of technical knowledge the user must have, operate the Contextual Object Detection System tends to be on a higher side because the user needs to know the environment to the configuration and dependency management. Small organizations without resources and an IT staff or non-technical users must confront deployment obstacles of technical systems. The application offers a user-friendly operation once it is installed, but the installation and the maintenance a hurdle that needs to be looked at, particularly the adopted system in compromised environment with limited technical knowledge.

5.6. Future Work

5.6.1. Augmented Reality Integration

The development of the next versions of the Contextual Object Detection system depends on its implementation within the enhanced reality systems. So, the setup transitioned from an analysis instrument to an interactive visual overlay technology. Future deployments are intended to provide immediate object detection and overlay information that can be seen by a mobile device through camera detection, and where objects are detected and labelled together with the object labels underneath and close real-world views. The potential of the combination of AR systems is to enable users to obtain educational information about real objects or to view the navigation of streets according to recognition markers, and to provide industrial-grade guidance when using visual prompts during an assembly process. With this innovation, the existing object detection capabilities would receive an upgrade in functionality, enabling significantly better user utility and performance.

5.6.2. Autonomous Vehicle and Safety Applications

Although the envisaged goal is far from the state-of-the-art today, the system developed in this project shows substantial potential for future automatic driving and safety systems. Future researchers can improve these automatically detected models for traffic elements , road conditions and obstacle identification purposes. If the real-time optimization processing could also be integrated into vehicles, it would provide a technology not only useful for collision avoidance systems but also for assisting drivers in identifying pedestrians and further developing improved assistance systems. The Nepali road safety scene provides a big enough space for simple driver warning solutions designed for mobile handsets to be safety enablers. Further model optimization and speed optimization, aside from other sensor system integration, will be needed to realize this application.

5.6.3. Advanced Vision-Language Models

The method is based on the SigLIP vision encoder for visual feature and captioning. We believe that future researcher should benefit from adopting the more advanced vision-language models VLMs to perform better understanding with natural interaction. By incorporating enhanced VLM functions into the system, questions focused on the context of the content of pictures can be made

to obtain the relevant responses. Such progress would transform the passive user viewing of detection outputs to an active visual dialogue experience. This will need big multimodal models with better prompting methods and the trend of natural dialogue development for visual understanding.

5.6.4. Assistive Technology for Visual Impairments

Another notable important future direction is the development of Contextual Object Detection systems as an assistive technology for individuals with visual impairment. The future work should be concentrated on achieving real-time feature description, as well as obstacle detection and text recognition, and navigation. The speech interface needs improvement with more models for automatically describing what is seen in scenes composed with the ability to recognize text in documents and sign captions as well. The system requires a great deal of emphasis on consistency with low latency and performance accuracy because users rely on the system for critical environmental information. Critical partnerships between software developers and visually disabled users, in conjunction with organizations to create truly effective implementations based on real-world necessities.

6. Chapter 6: References

Adam Paszke, S. G. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *ArXiv*.

Alexey Dosovitskiy, L. B. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ArXiv*.

Beigel, K. (2025, 04 28). *RealtimeSTT*. Retrieved from GitHub: https://github.com/KoljaB/RealtimeSTT

Chainer. (2025, 04 28). *Bridge the gap between algorithms and implementations of deep learning*. Retrieved from Chainer: https://chainer.org/

Das, S. (2024, October 7). *What is System Testing? (Examples, Use Cases, Types)*. Retrieved from BrowserStack: https://www.browserstack.com/guide/what-is-system-testing

DeepSeek. (2025, 04 25). *DeepSeek: Into the unknown*. Retrieved from DeepSeek: https://www.deepseek.com/

Google Deepmind. (2025, 04 25). *Gemini*. Retrieved from Google Deepmind: https://deepmind.google/technologies/gemini/

Hugging Face. (2025, 04 23). *Philosophy*. Retrieved from Hugging Face: https://huggingface.co/docs/transformers/en/philosophy

IBM. (2025). *IBM Watson to watsonx*. Retrieved from IBM: https://www.ibm.com/watson

Keita, Z. (2025, 04 28). *An Introduction to Using Transformers and Hugging Face*. Retrieved from DataCamp: https://www.datacamp.com/tutorial/an-introduction-to-using-transformers-and-hugging-face

Khatiwada, D. (2025, February 10). *GadgetByte*. Retrieved from All about the latest National AI Policy: Key elements and challenges ahead: https://www.gadgetbytenepal.com/national-ai-policy/

Mozilla Foundation. (2025, 04 28). *Django introduction*. Retrieved from Django Introduction: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Django/Introduction

Nvidia. (2025, 03 9). *PyTorch*. Retrieved from Nvidia: https://www.nvidia.com/en-us/glossary/pytorch/

OpenAI. (2025, 04 25). *Introducing ChatGPT*. Retrieved from OpenAI: https://openai.com/index/chatgpt/

Redis . (2025, 04 28). *Solutions Library*. Retrieved from Redis: https://redis.io/solutions/

Redis. (2023). *What is Redis?* Retrieved from GitHub: https://github.com/redis/redis

Sara Abdali, R. A. (2024). Decoding the AI Pen: Techniques and Challenges in Detecting AI-Generated Text. *ArXiv*.

Shilong Liu, Z. Z. (2023). Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection. *ArXiv*, abs/2303.05499.

Suwal, P. (2024, August 14). *Artificial Intelligence in Nepal's Education Sector*. Retrieved from Nepal Economic Forum: https://nepaleconomicforum.org/artificial-intelligence-in-nepals-education-sector/

The Kathmandu Post. (2025, January 7). *Not ready for AI*. Retrieved from The Kathmandu Post: https://kathmandupost.com/editorial/2025/01/07/not-ready-for-ai

Theano. (2018). Theano. Retrieved from https://github.com/Theano/Theano

Tricentis Testim. (2019, September 04). *Scrum Testing: A Detailed Guide to Testing on an Agile Team*. Retrieved from Tricentis Testim: https://www.testim.io/blog/scrum-testing-guide/

UN Trade and Development. (2025, April 07). *AI market projected to hit $4.8 trillion by 2033, emerging as dominant frontier technology*. Retrieved from UNCTAD: https://unctad.org/news/ai-market-projected-hit-48-trillion-2033-emerging-dominant-frontier-technology#:~:text=Artificial%20intelligence%20(AI)%20is%20fast,increase%20in%20just%20a%20decade.

Verma, V. (2024, March 30). *Introducing Moondream2: A Tiny Vision-Language Model*. Retrieved from Analytics Vidya: https://www.analyticsvidhya.com/blog/2024/03/introducing-moondream2-a-tiny-vision-language-model/

Yuhang Zang, W. L. (2024). Contextual Object Detection with Multimodal Large Language Models. *ArXiv*, abs/2305.18279.

Zhong-Qiu Zhao, P. Z.-t. (2018). Object Detection With Deep Learning: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 3212-3232.

7. Chapter 7: Appendix

7.1. Appendix 1: Pre-Survey Form

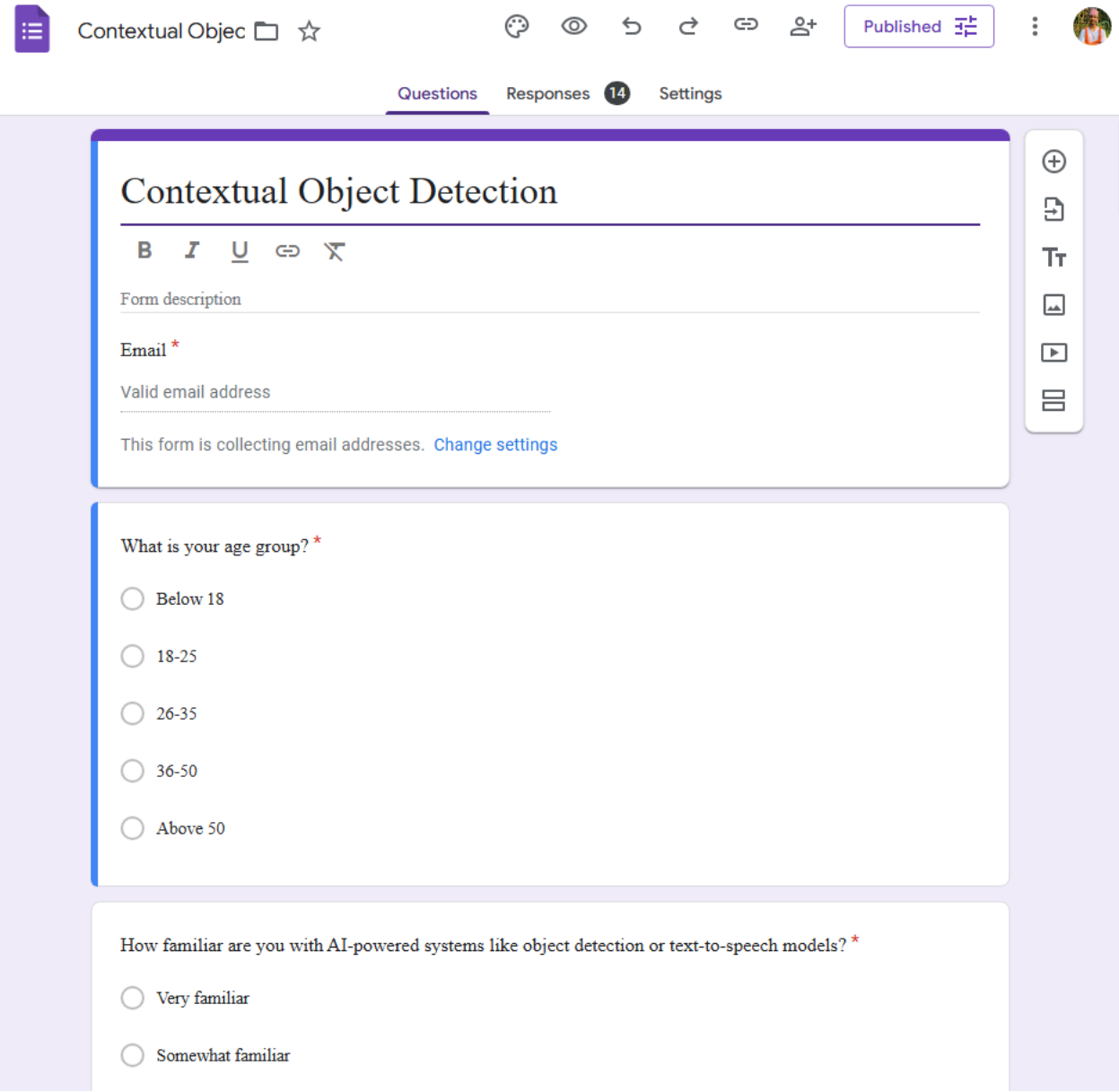


Figure 158: Pre survey form-1

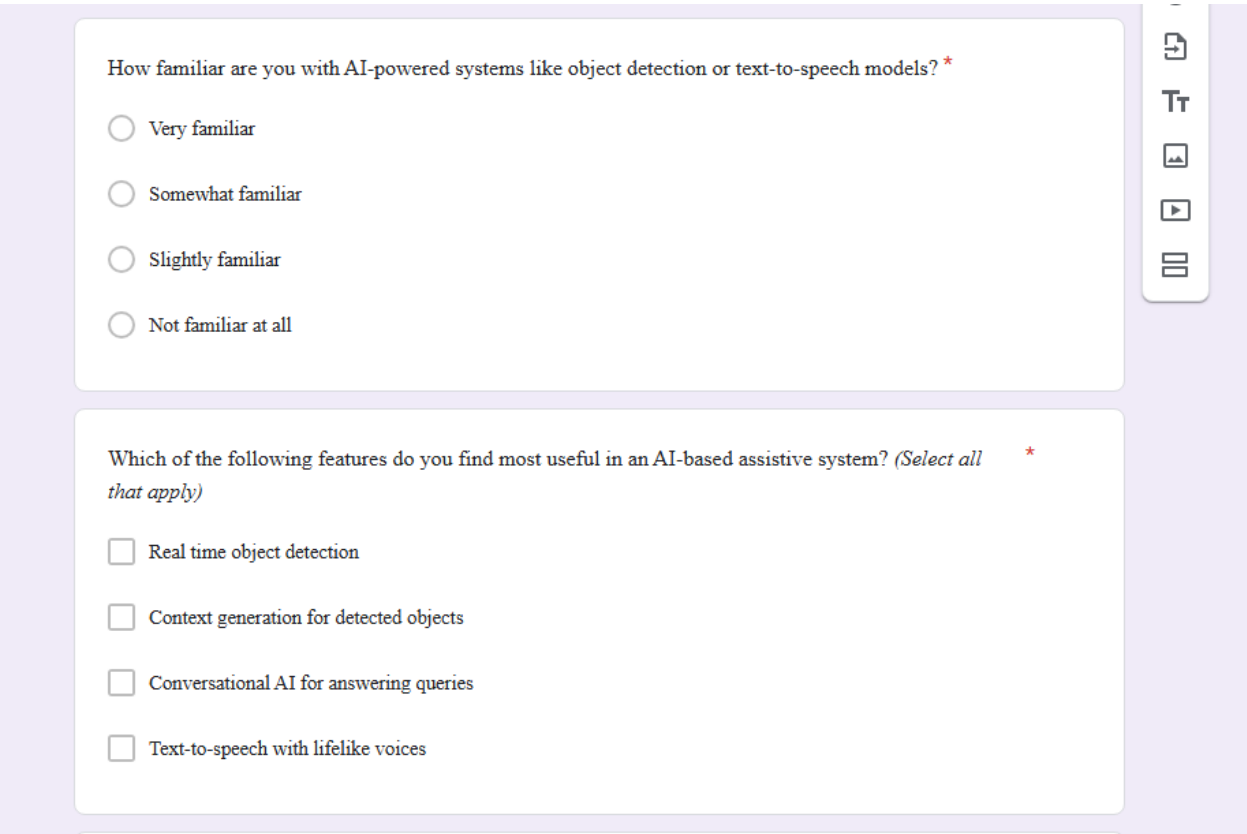


Figure 159: Pre survey form-2

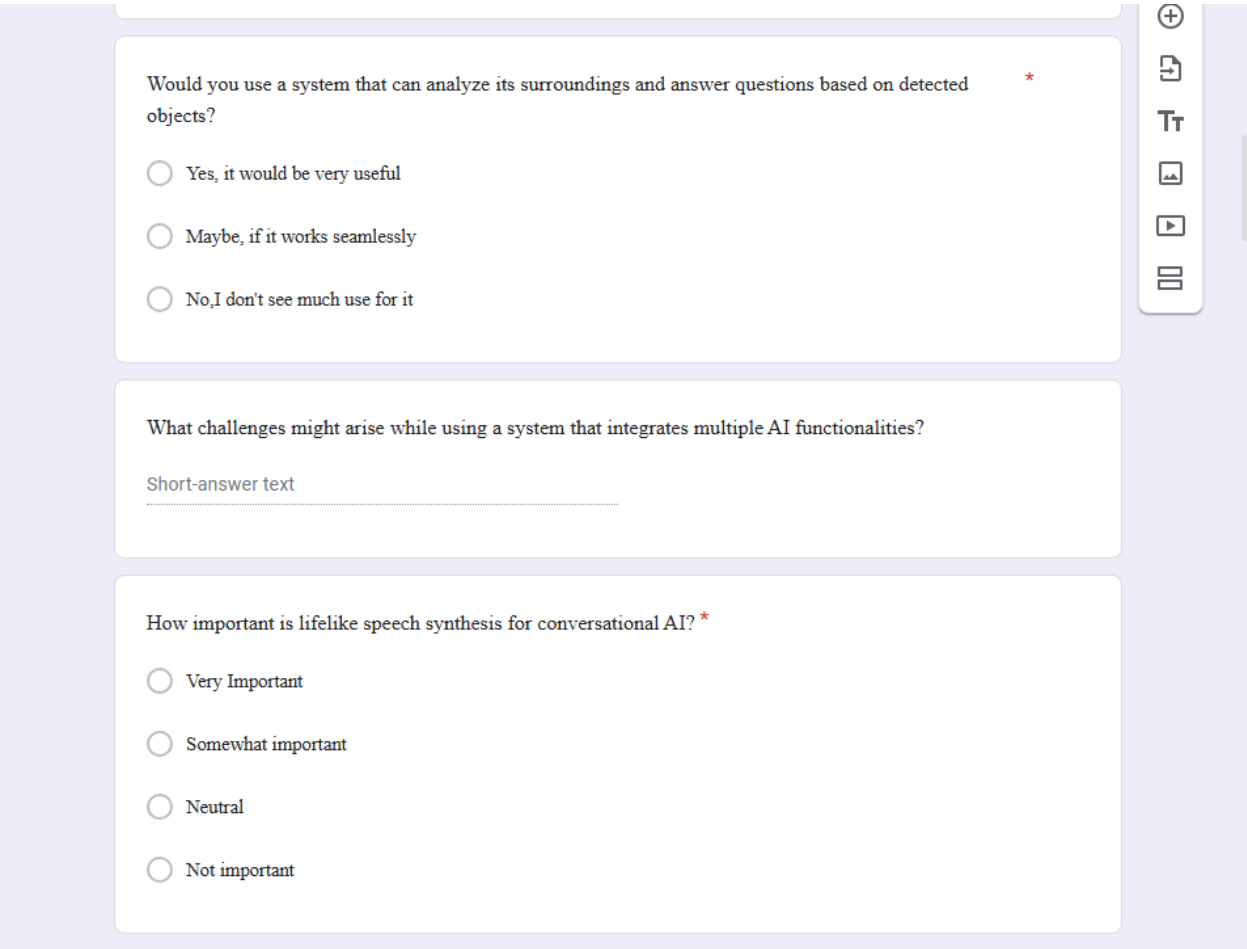


Figure 160: Pre survey form-3

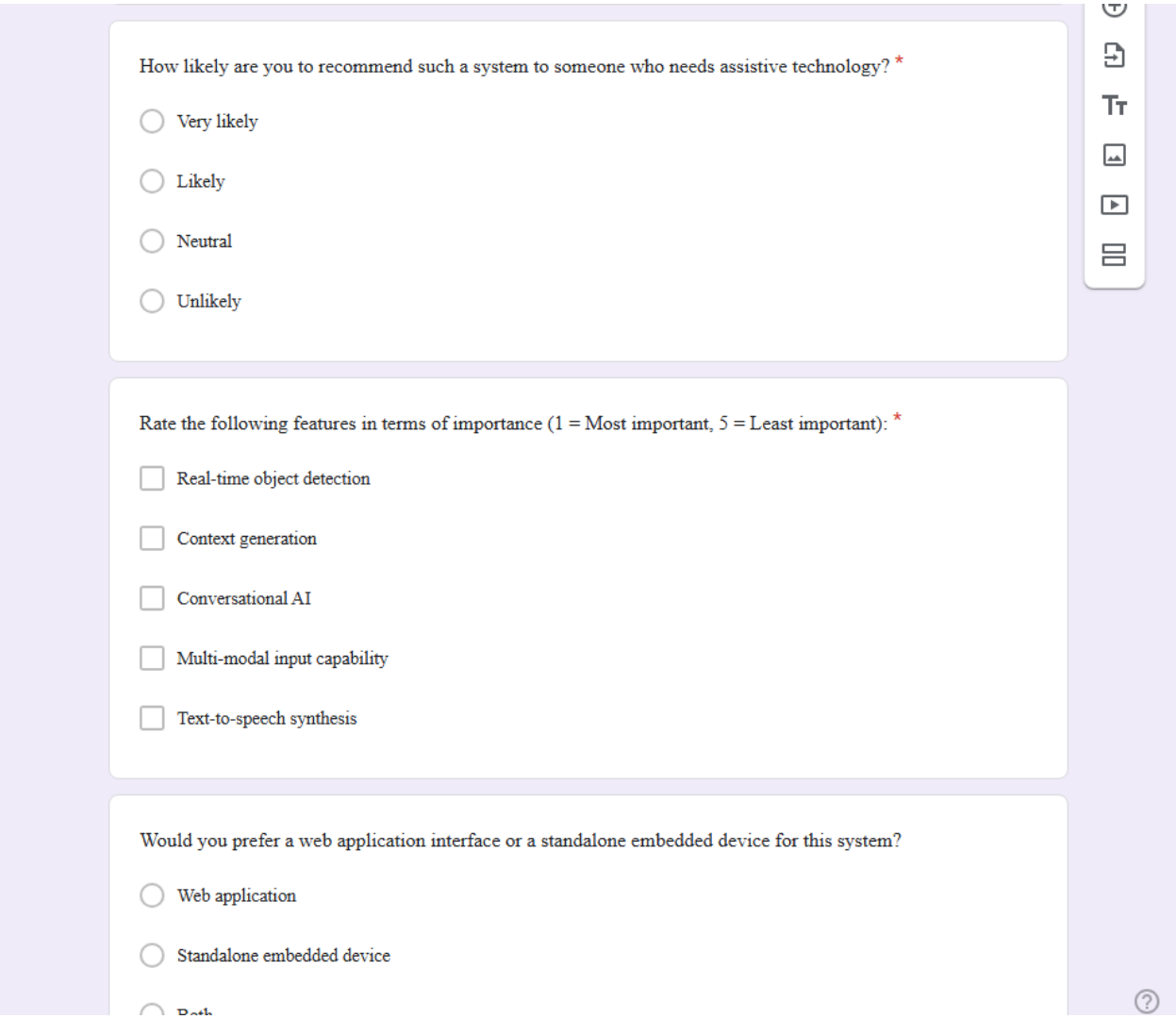


Figure 161: Pre survey form-4

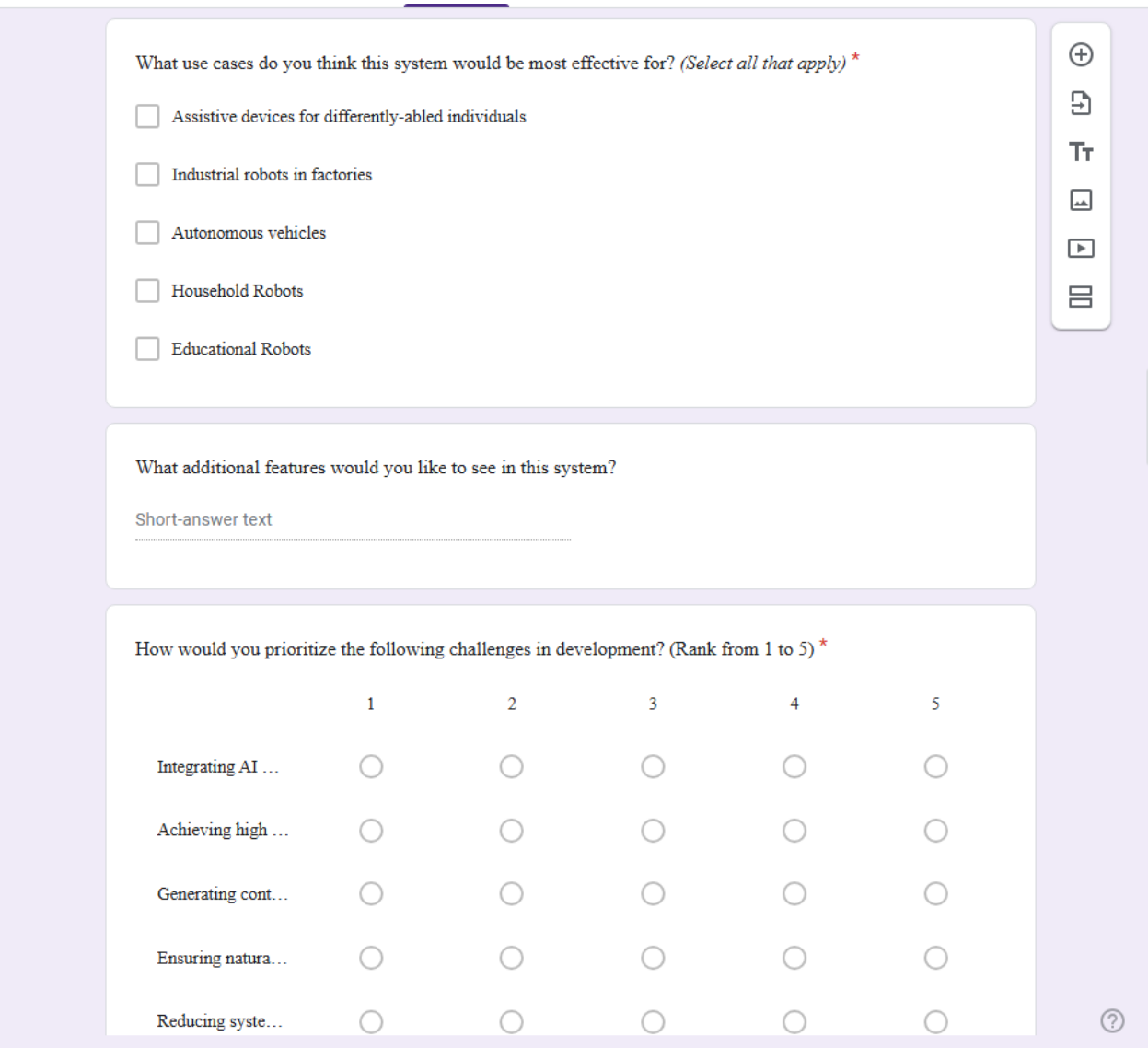


Figure 162: Pre survey form-6

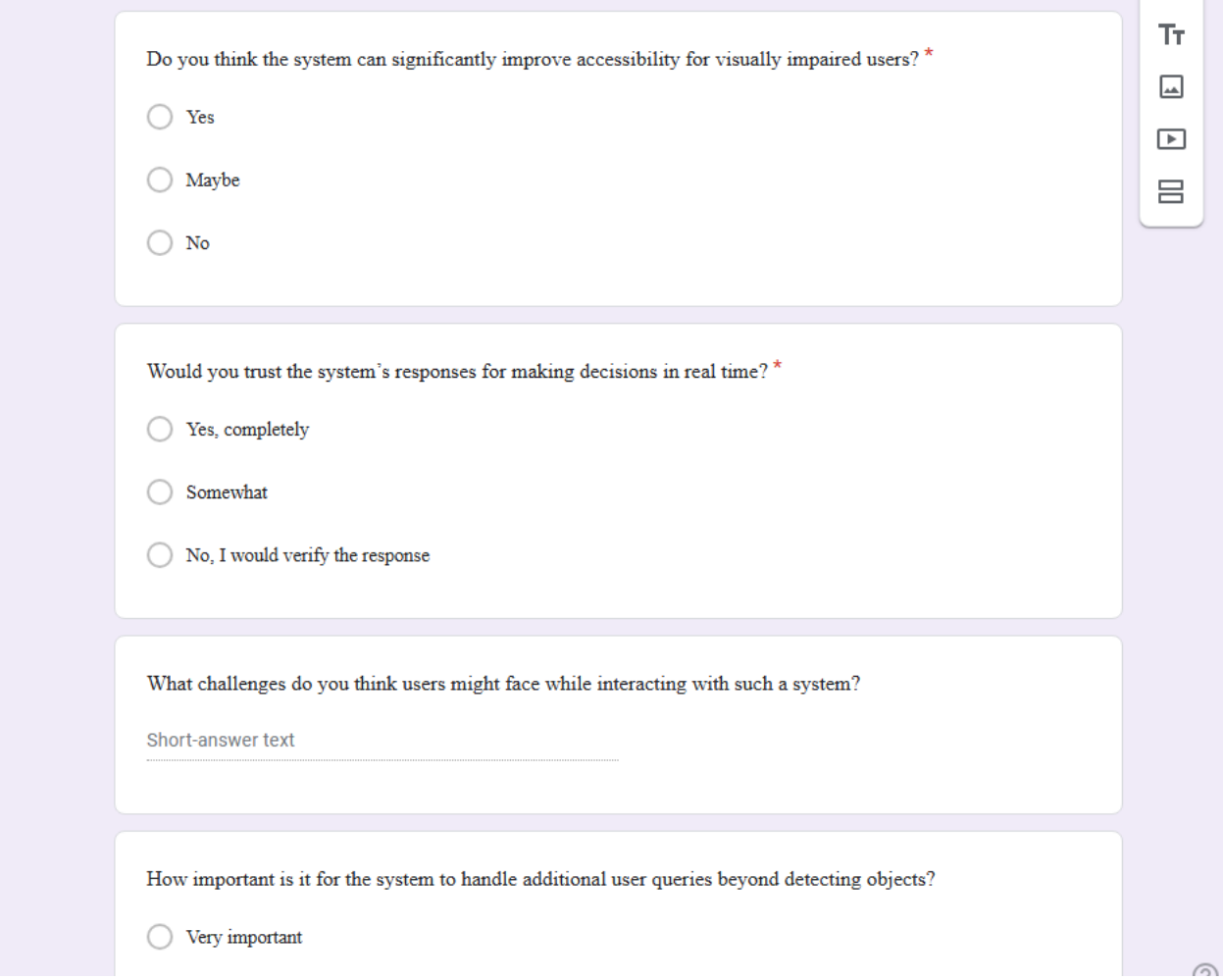


Figure 163: Post survey form-7

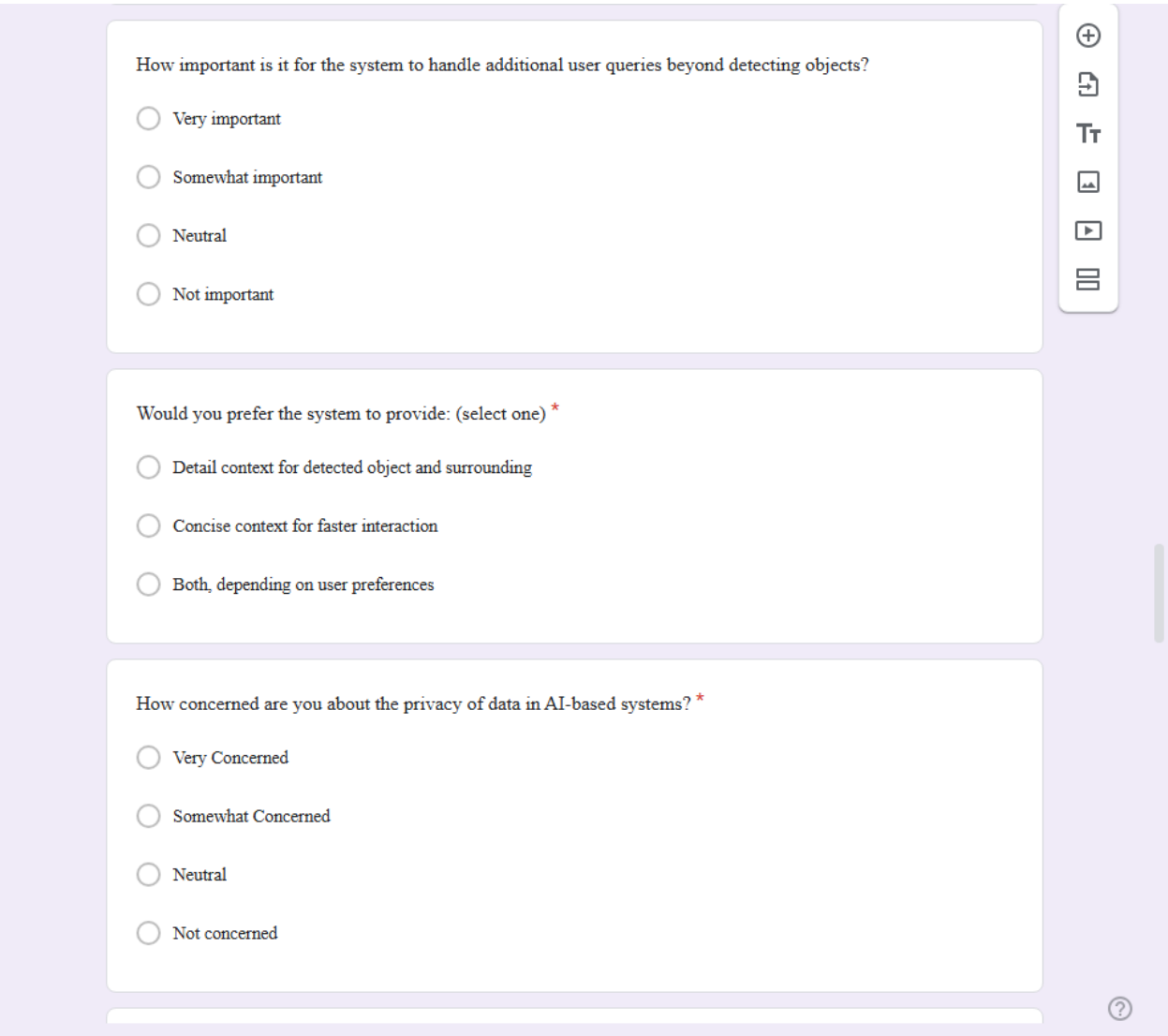


Figure 164: Post survey form-8

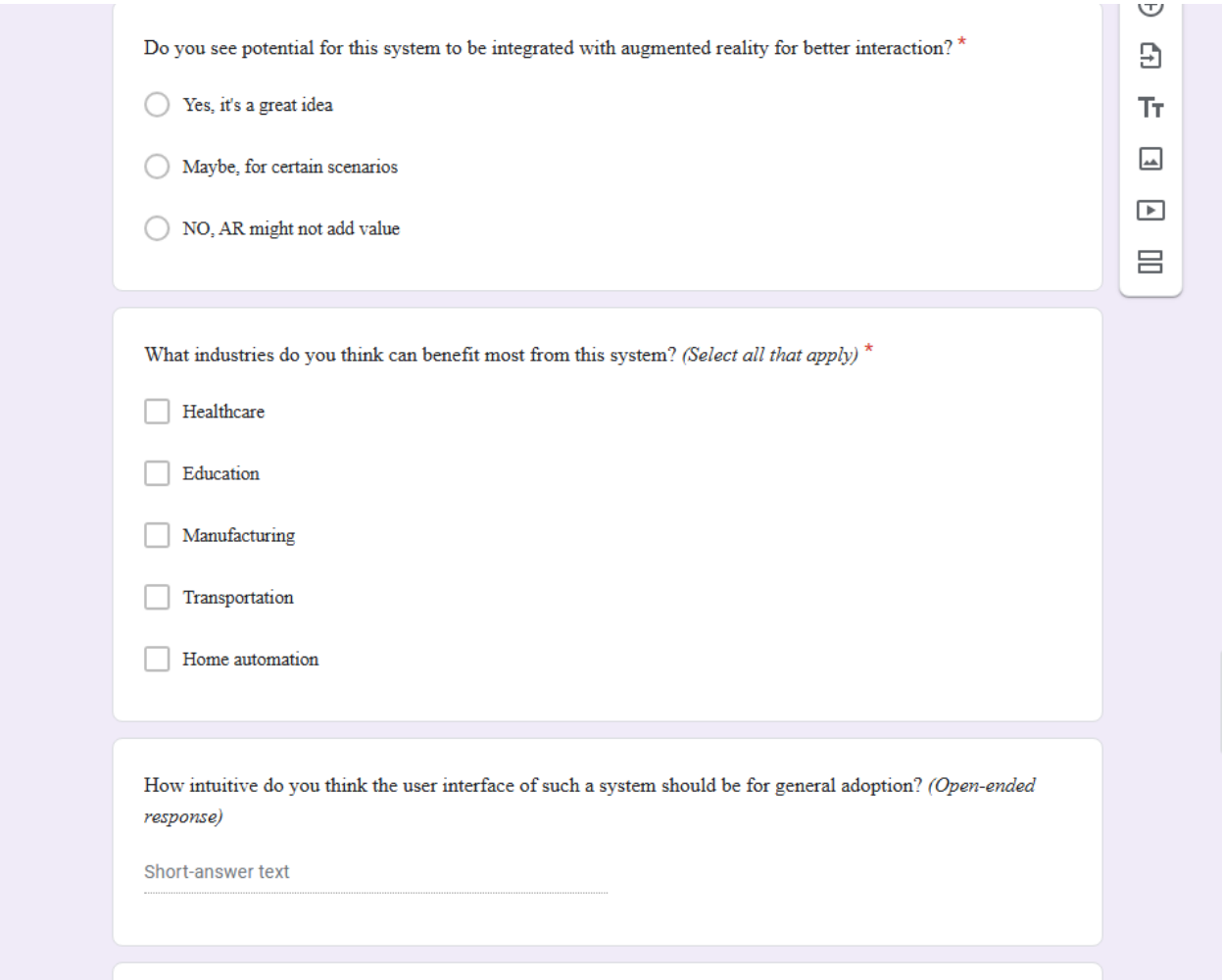


Figure 165: Pre survey form-9

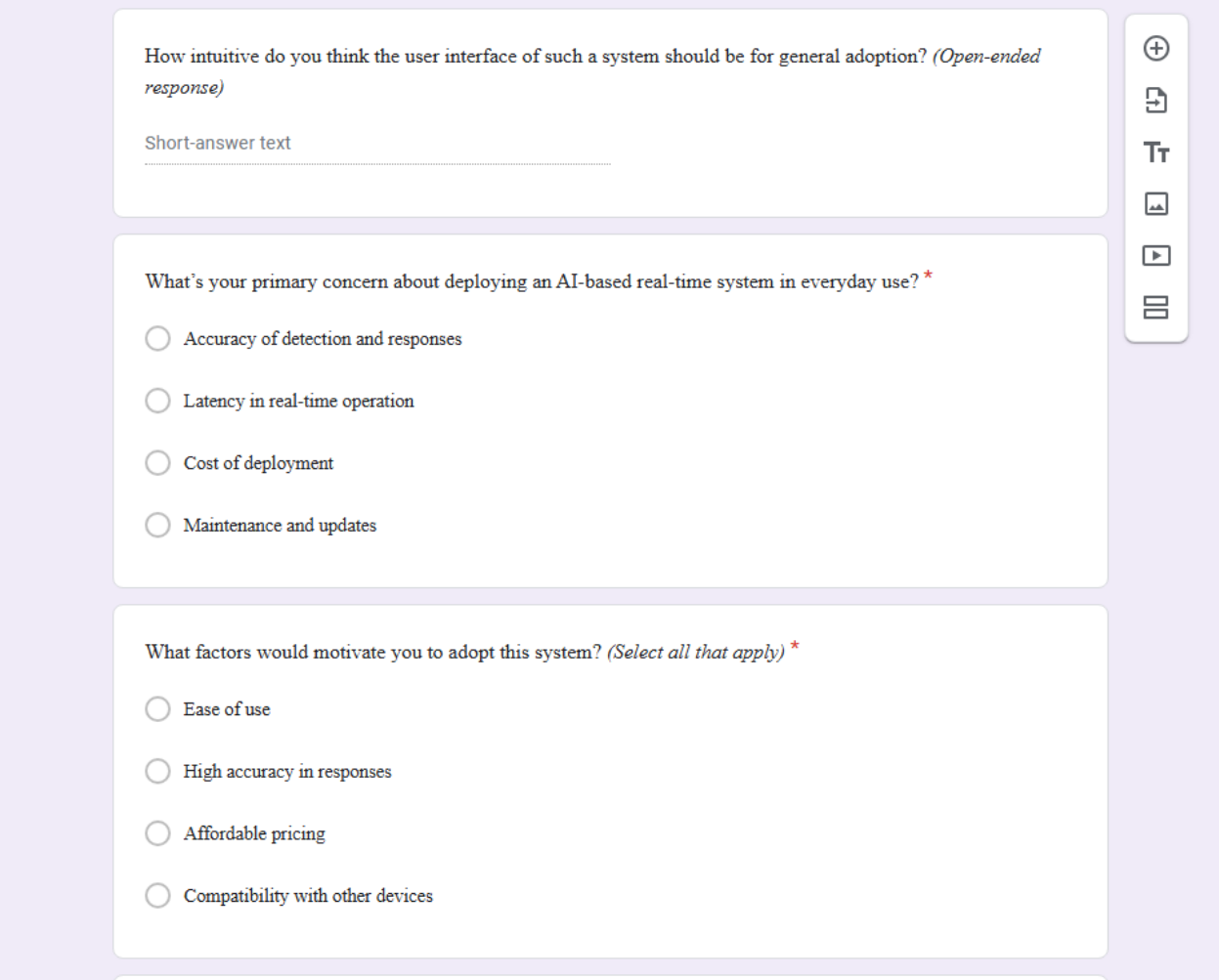


Figure 166: Pre survey form-10

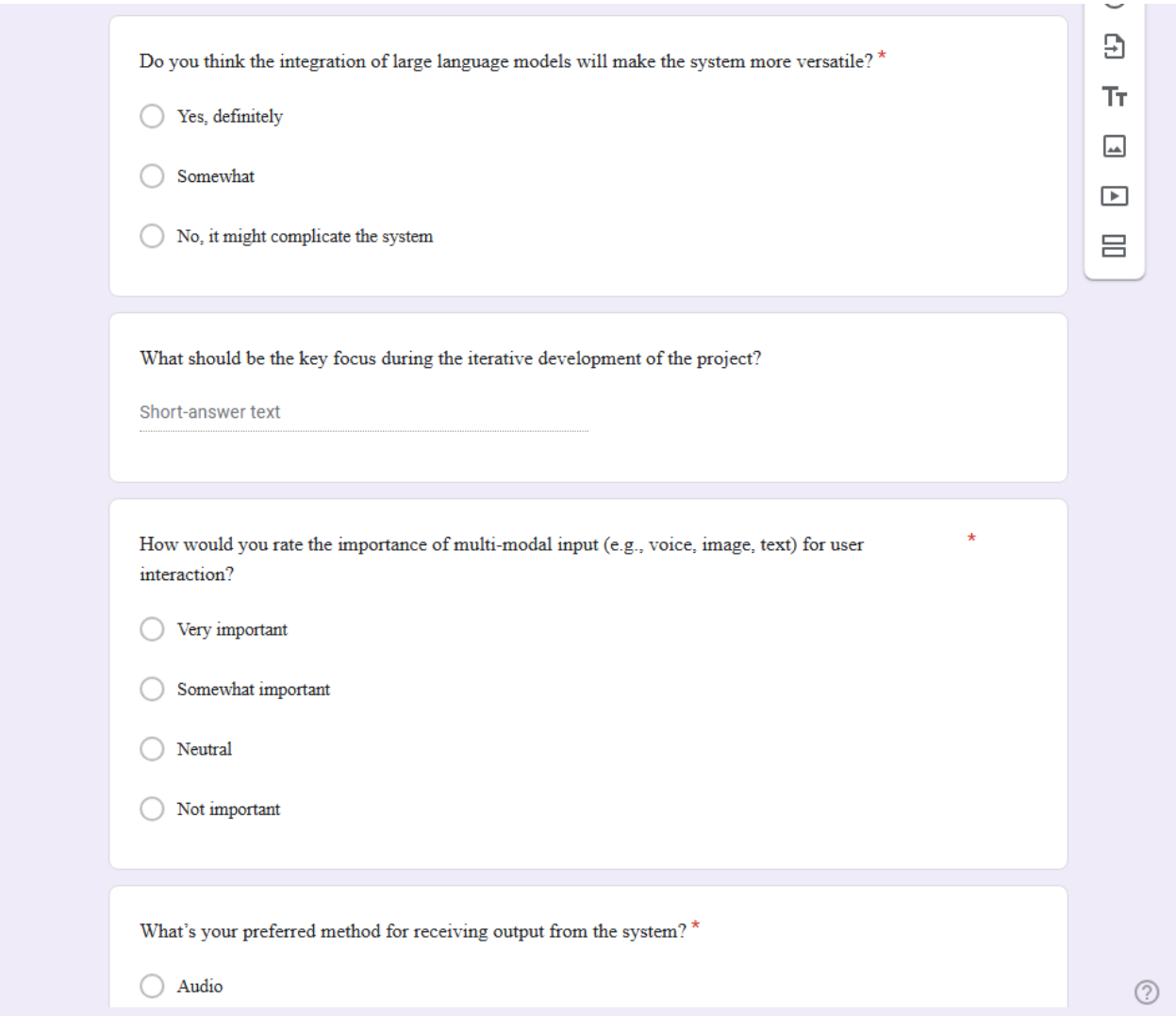


Figure 167: Pre survey form-11

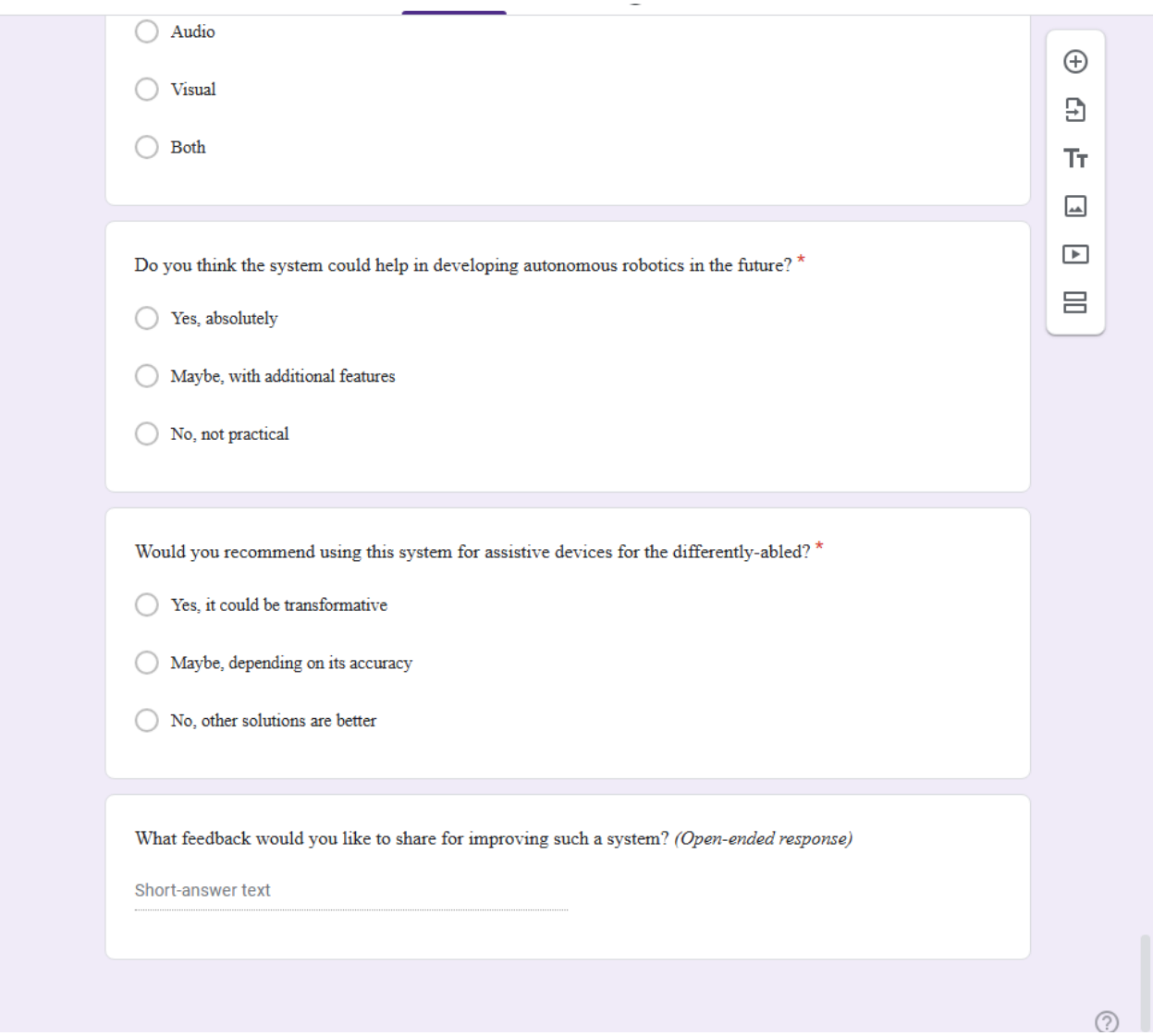


Figure 168: Pre survey form-12

7.2. Appendix 2: Pre-Survey Results

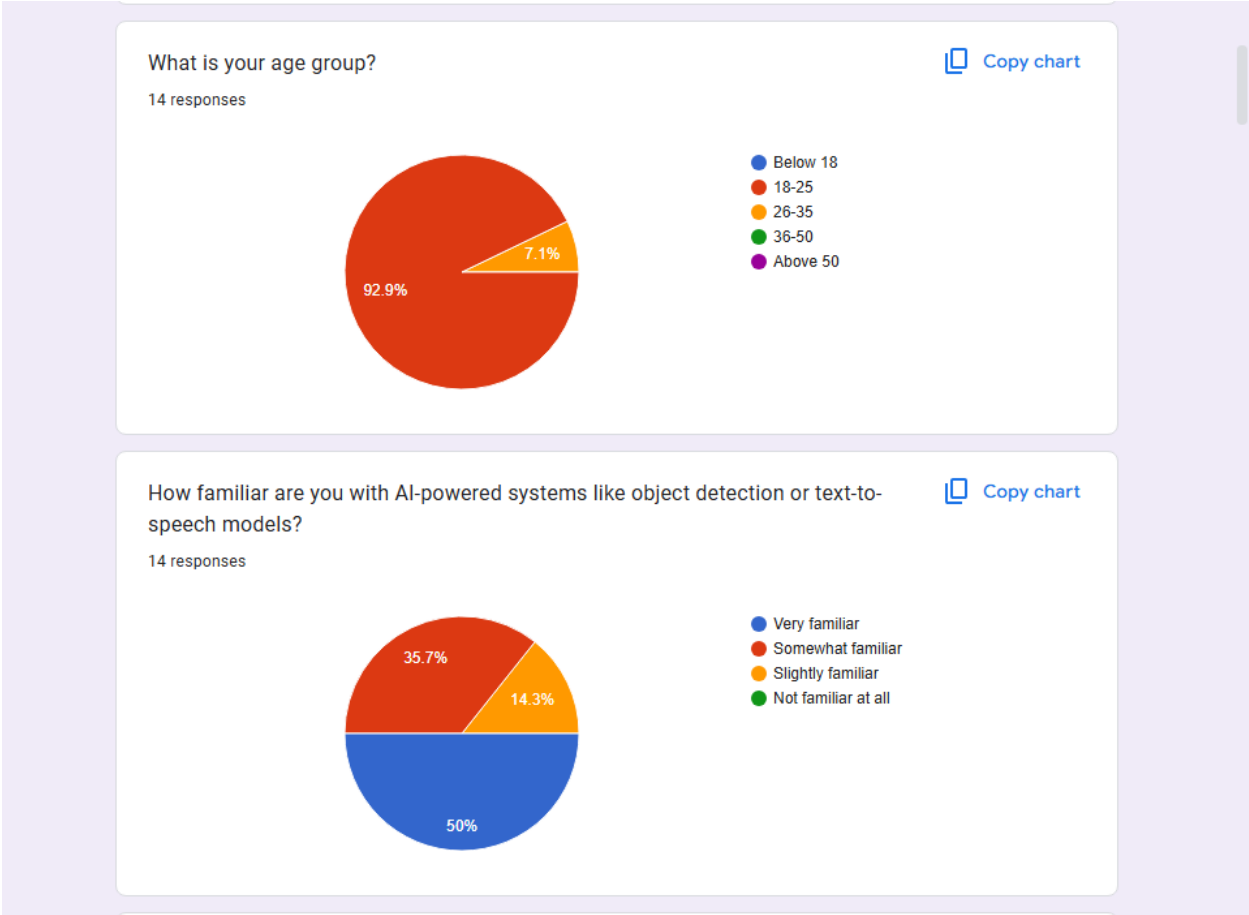


Figure 169: Pre survey result-1

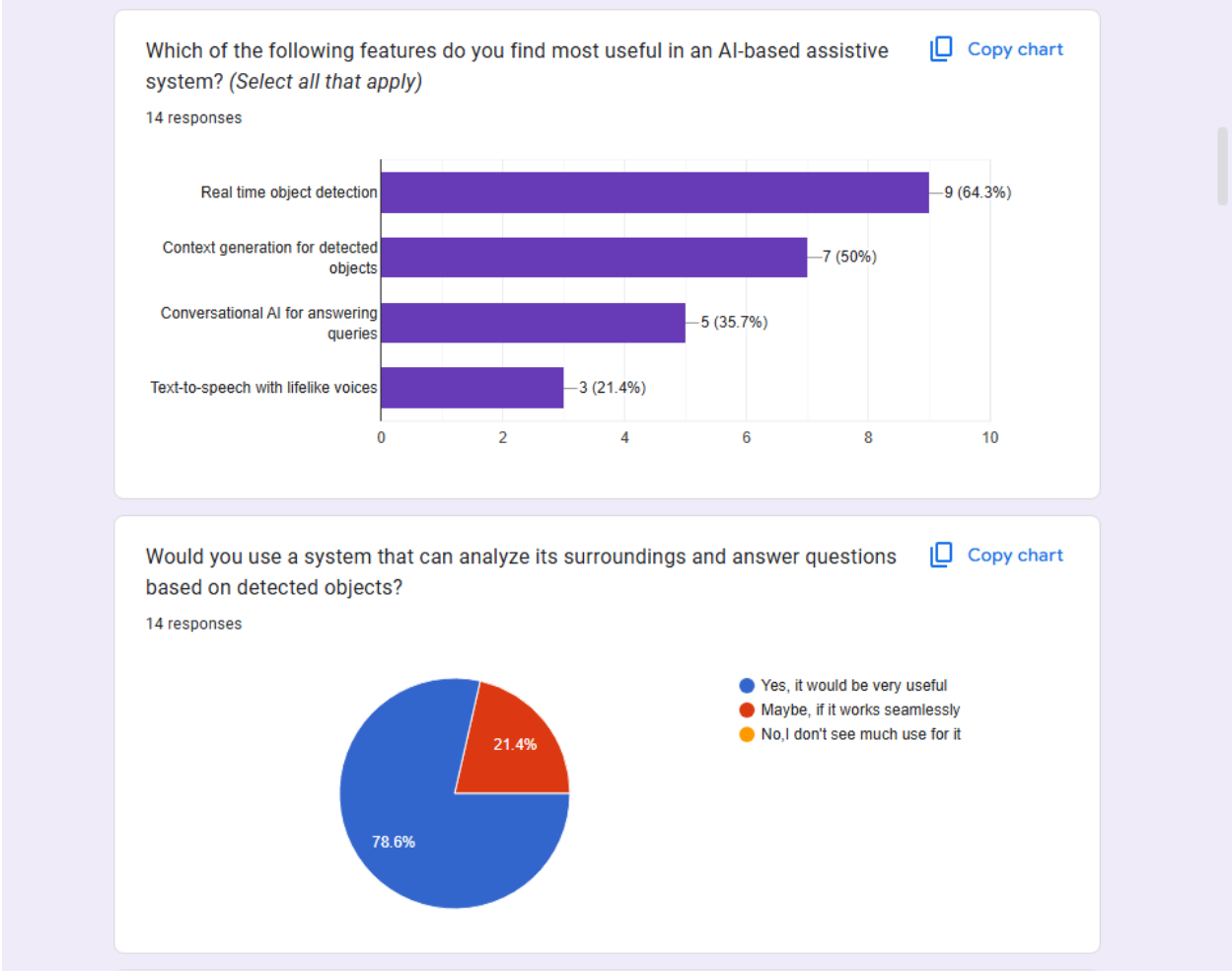


Figure 170: Pre survey results-1

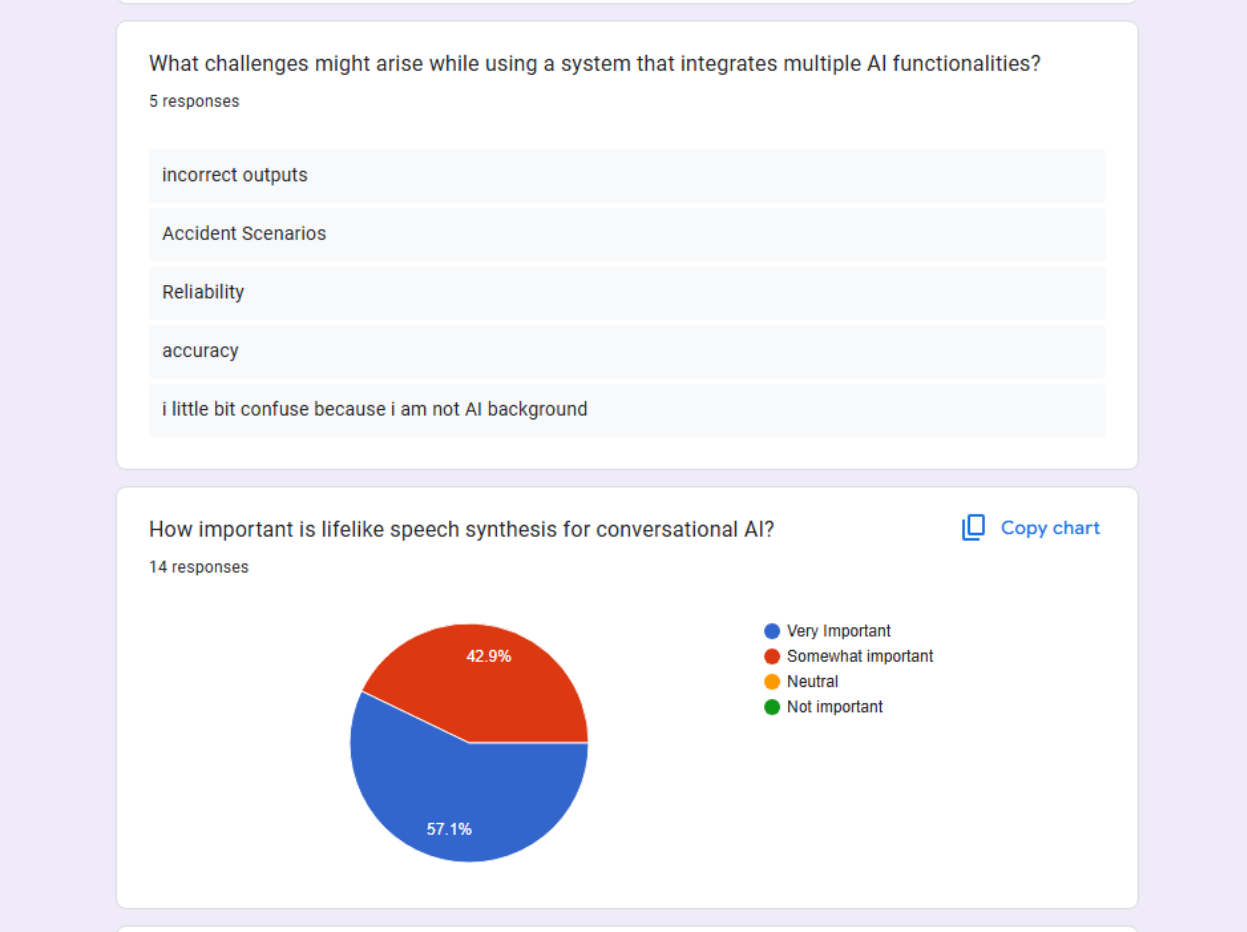


Figure 171: Pre survey results-3

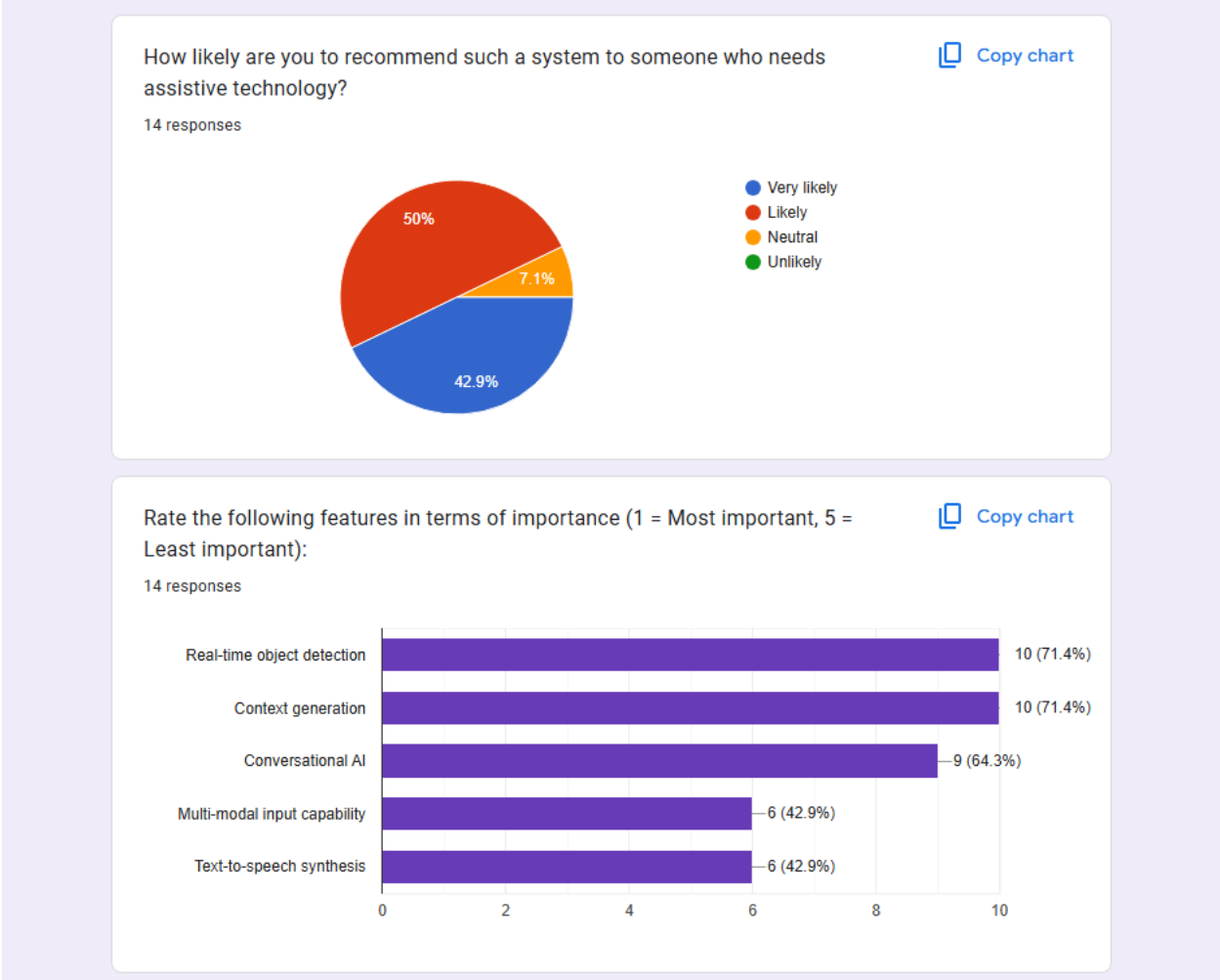


Figure 172: Pre-survey results-4

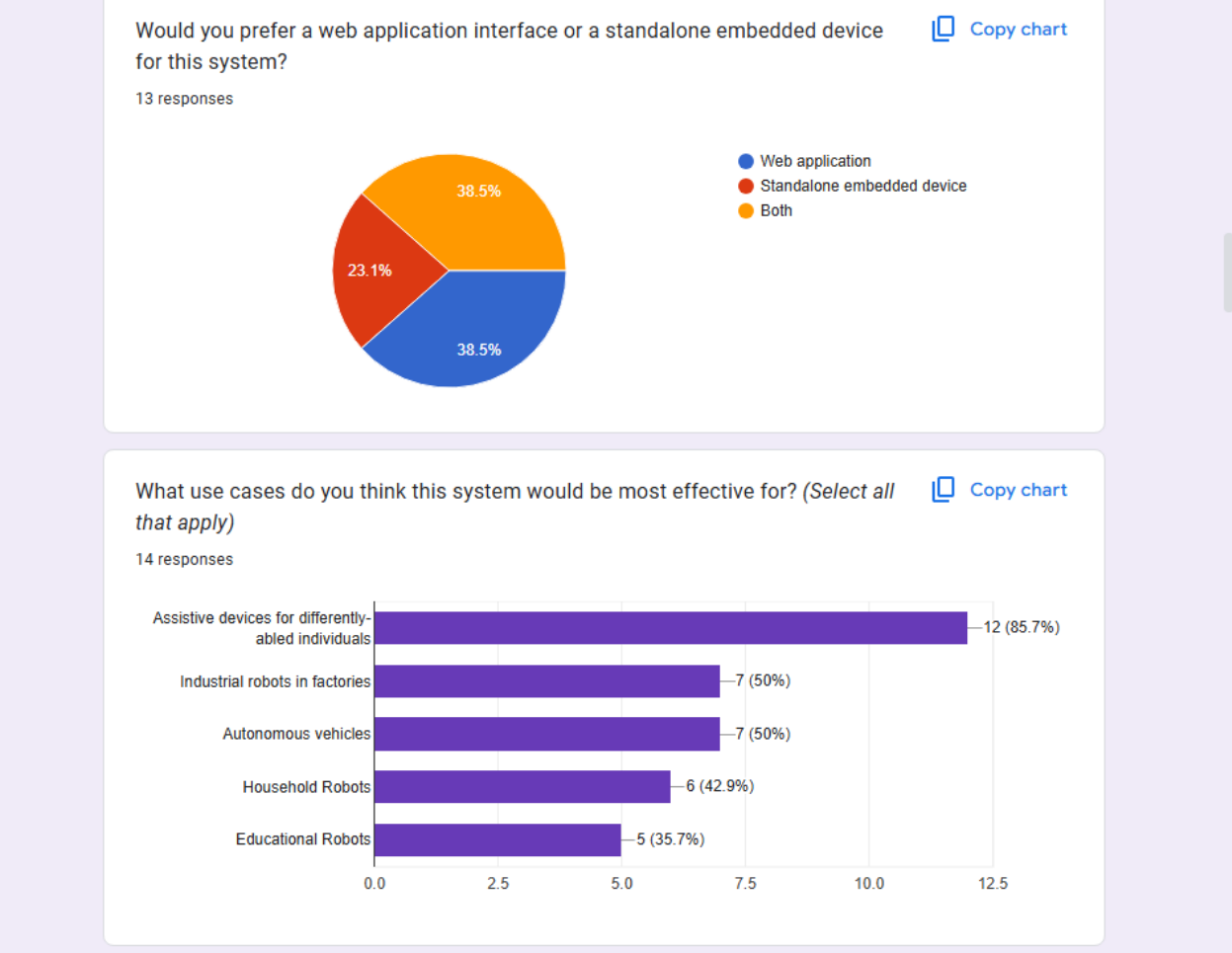


Figure 173: Pre survey results-6

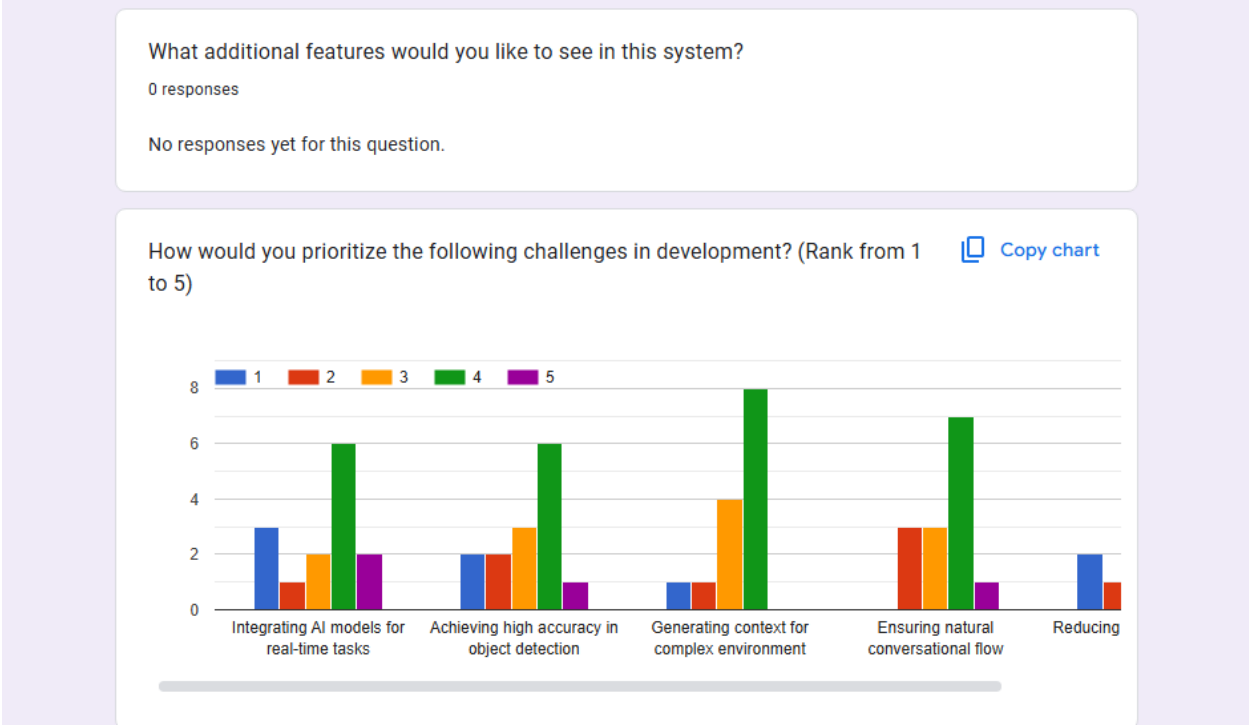


Figure 174: Pre survey results-7

Figure 175: Pre survey results-8

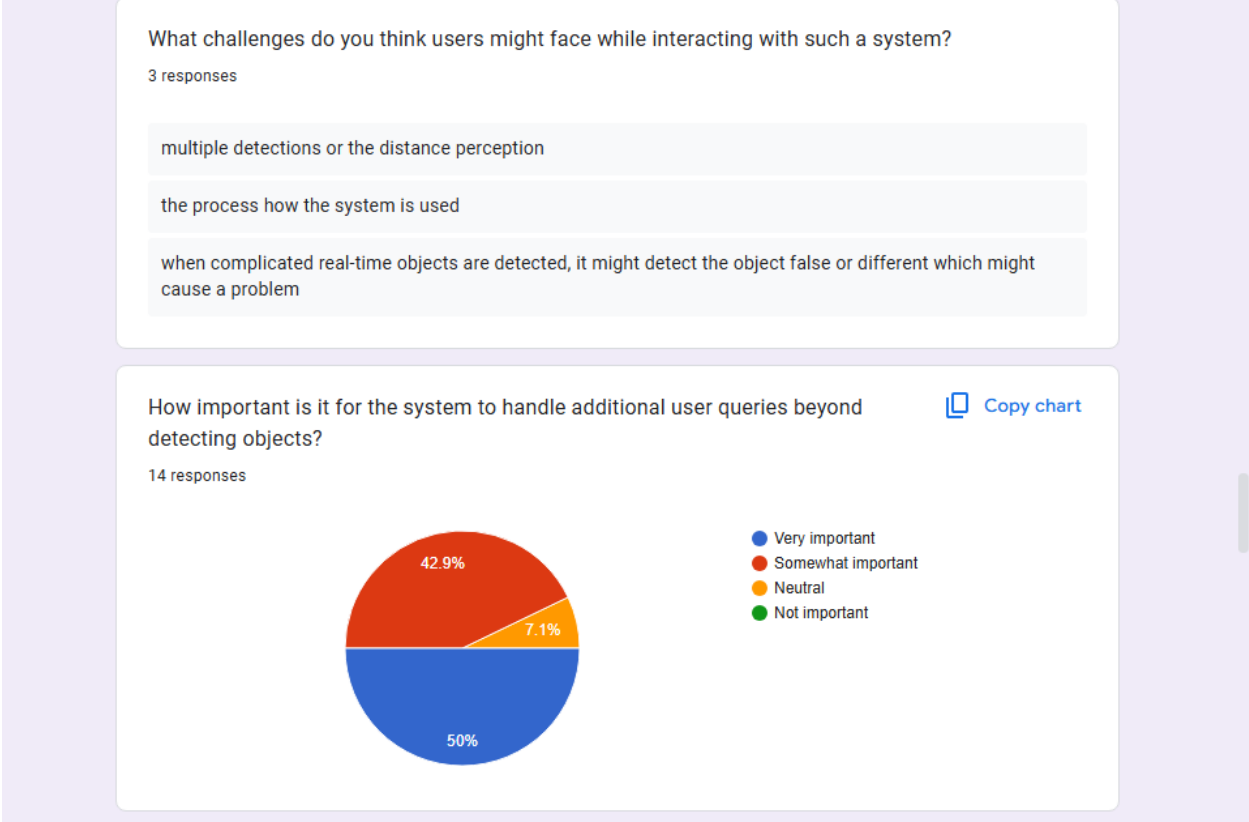
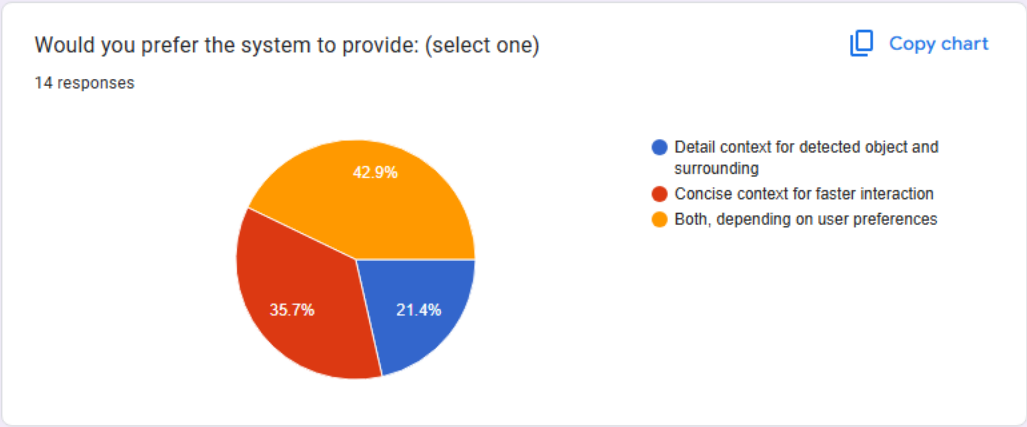


Figure 176: Pre survey results-9



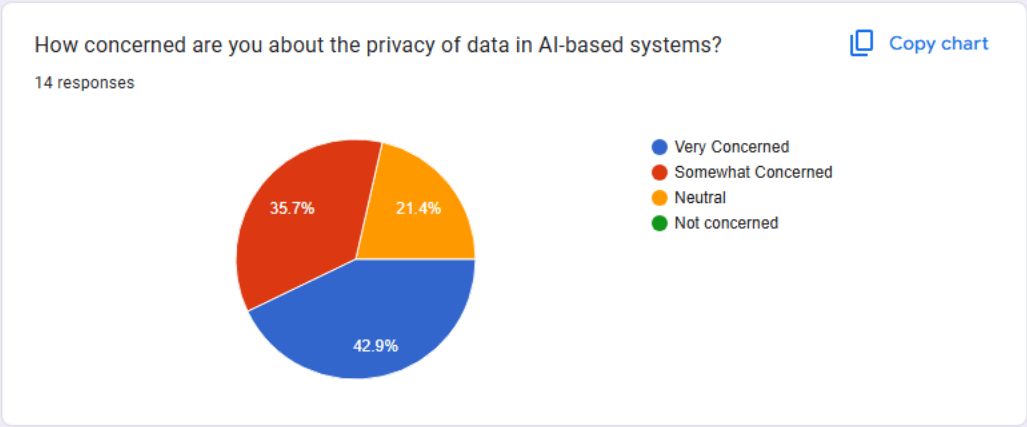


Figure 177: Pre survey results-10

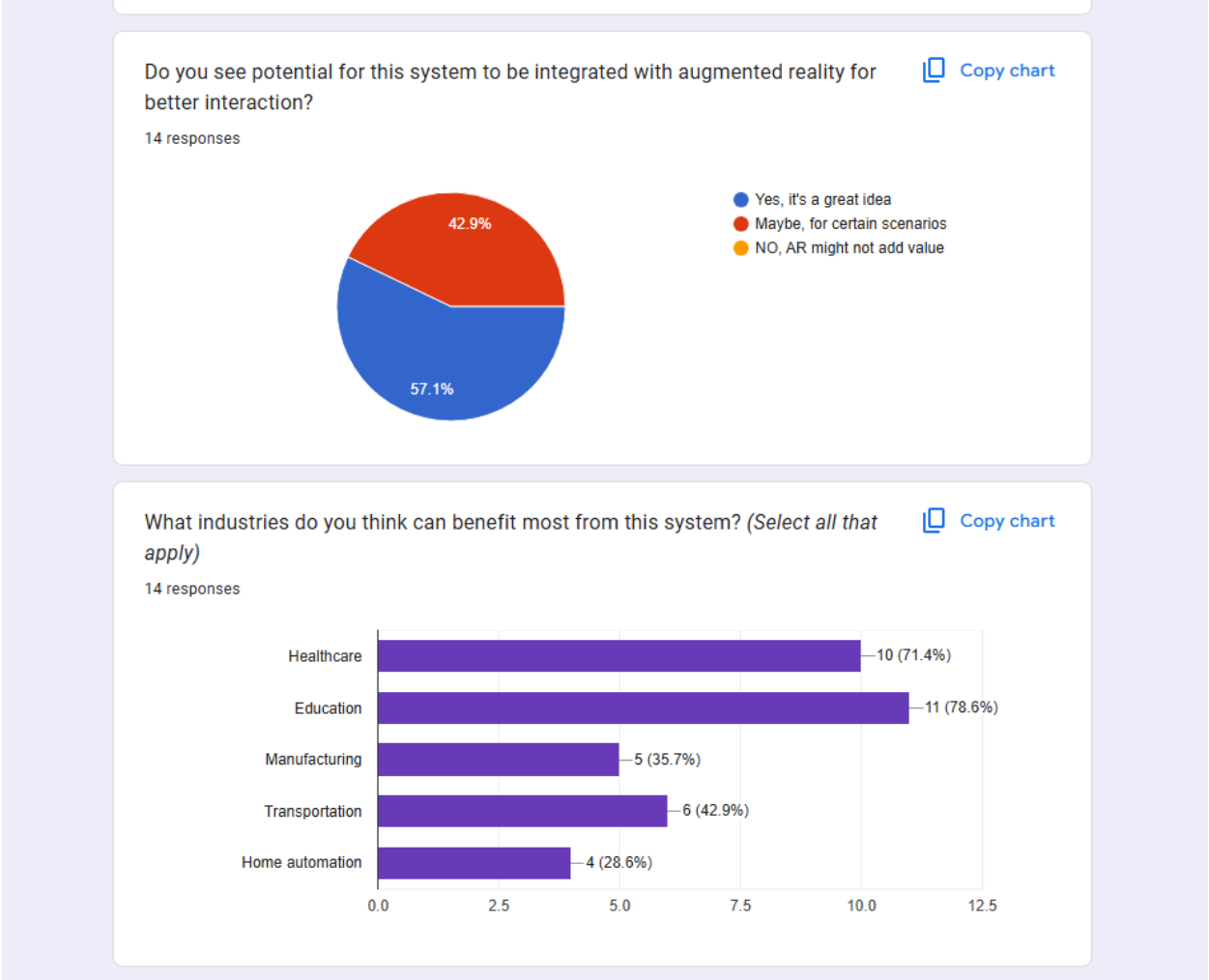


Figure 178: Pre survey results-11

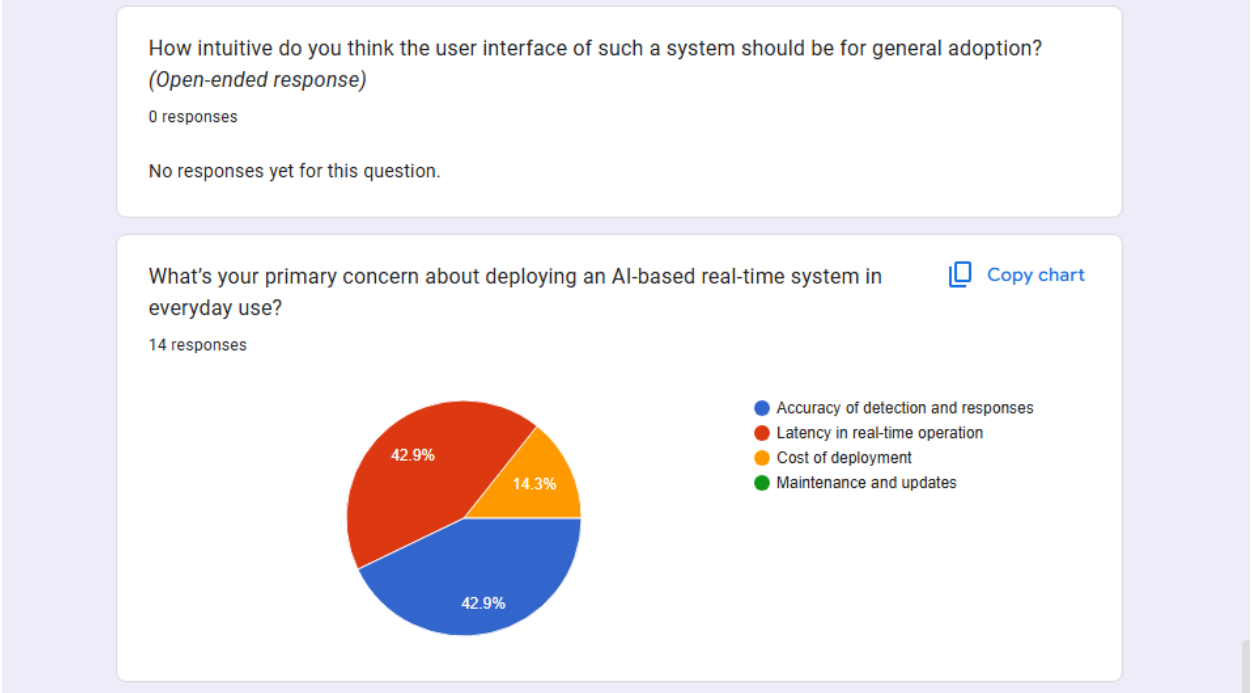


Figure 179: Pre survey results-12

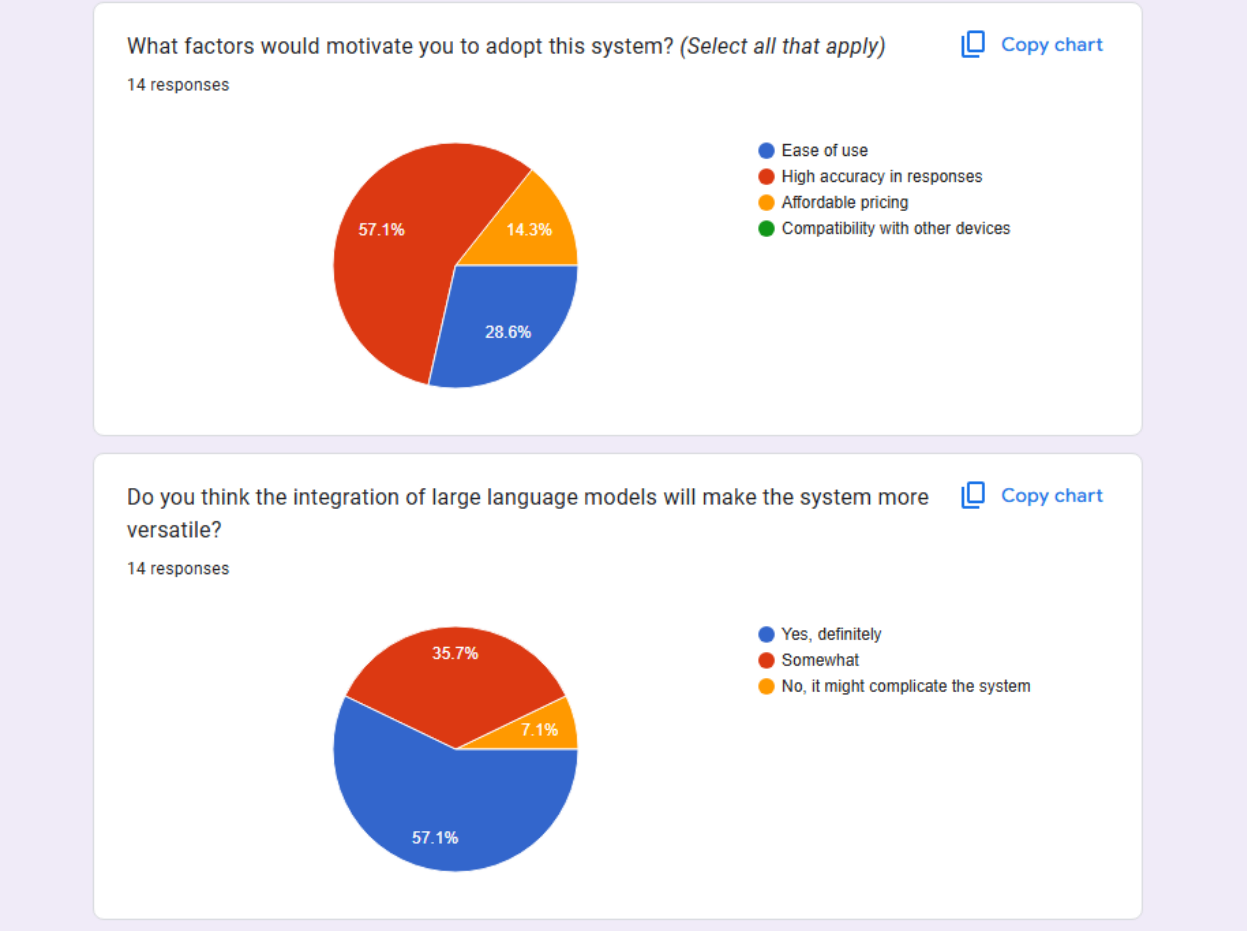


Figure 180 : Pre survey results-13

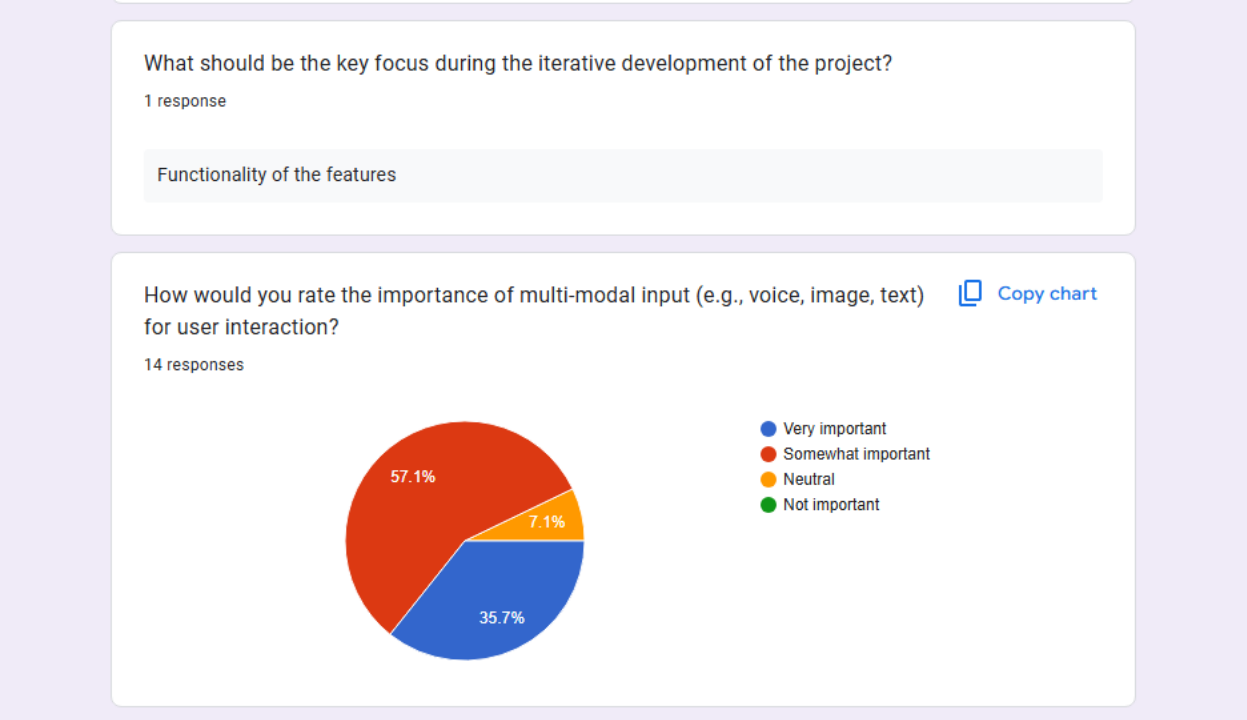


Figure 181: Pre survey results-13

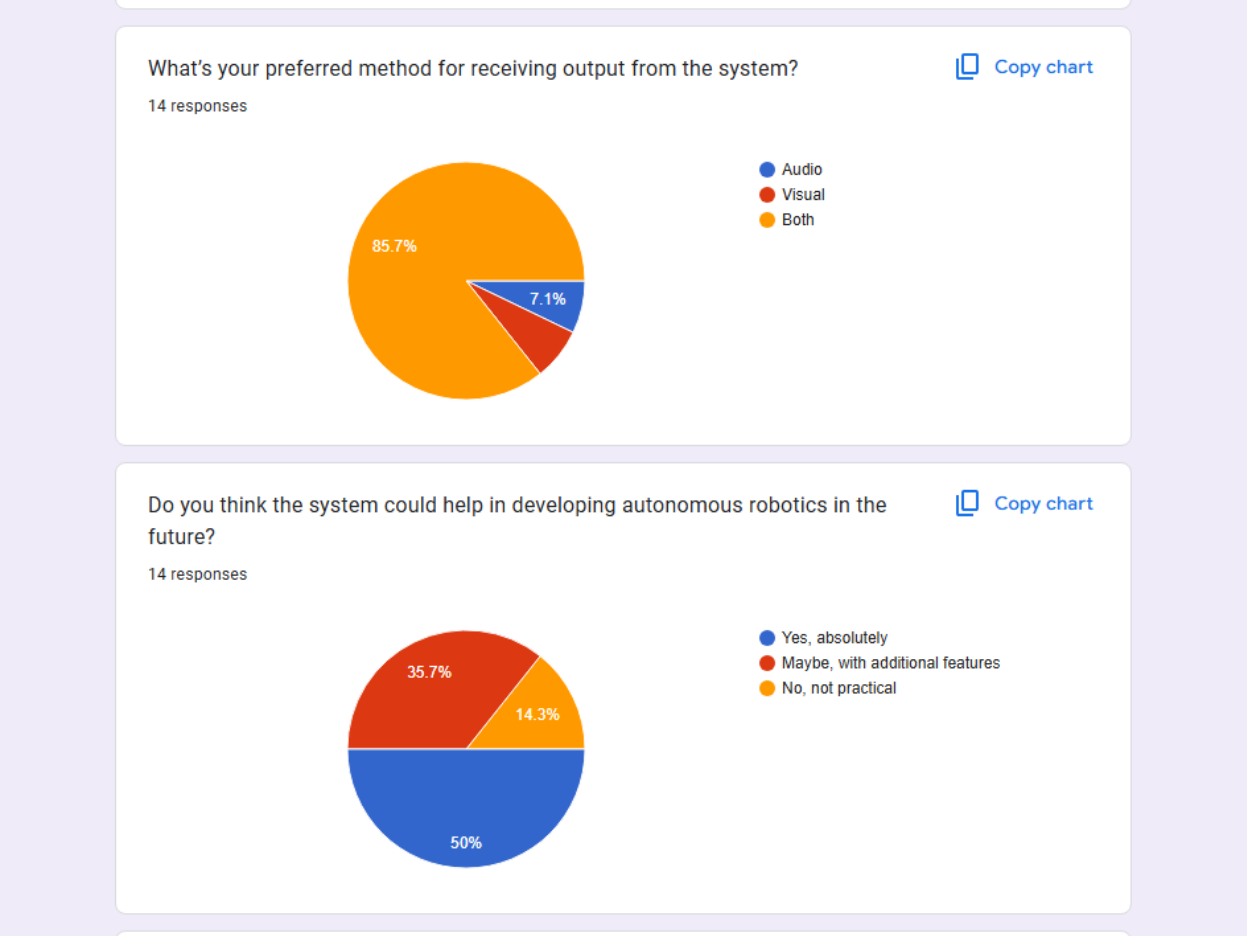


Figure 182: Pre survey results-14

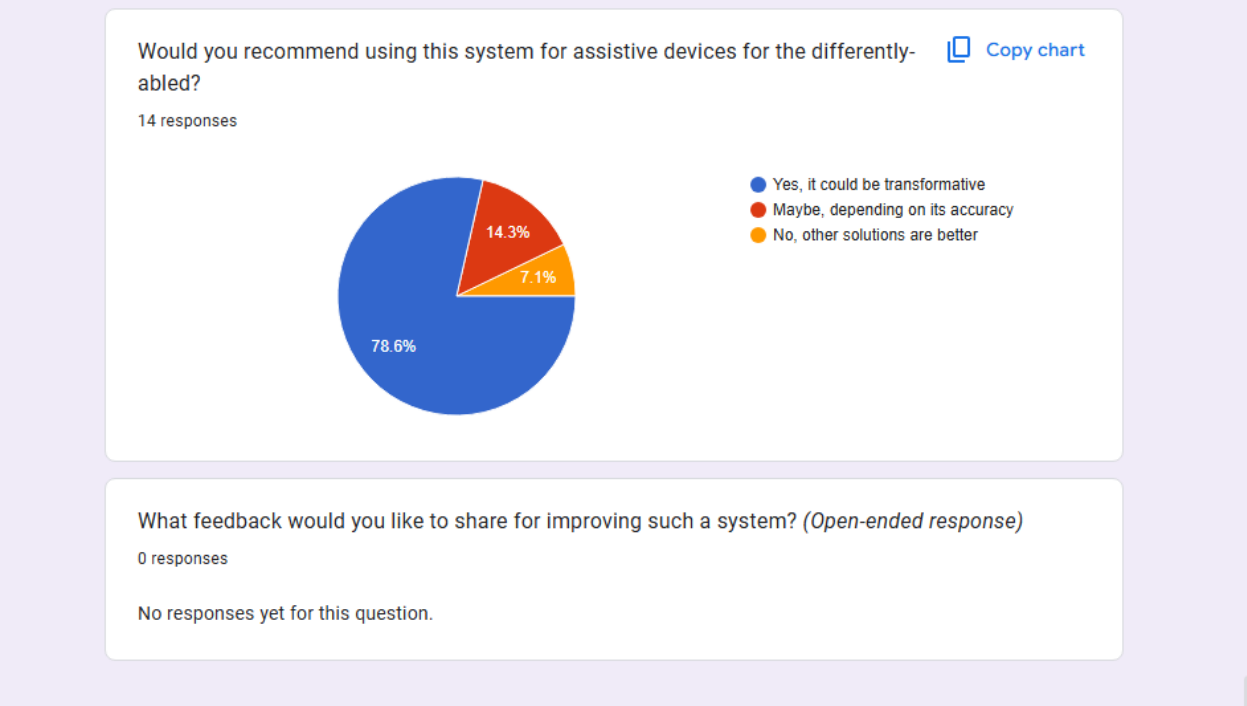


Figure 183: Pre survey results-15

7.3. Appendix 3: Post Survey Form

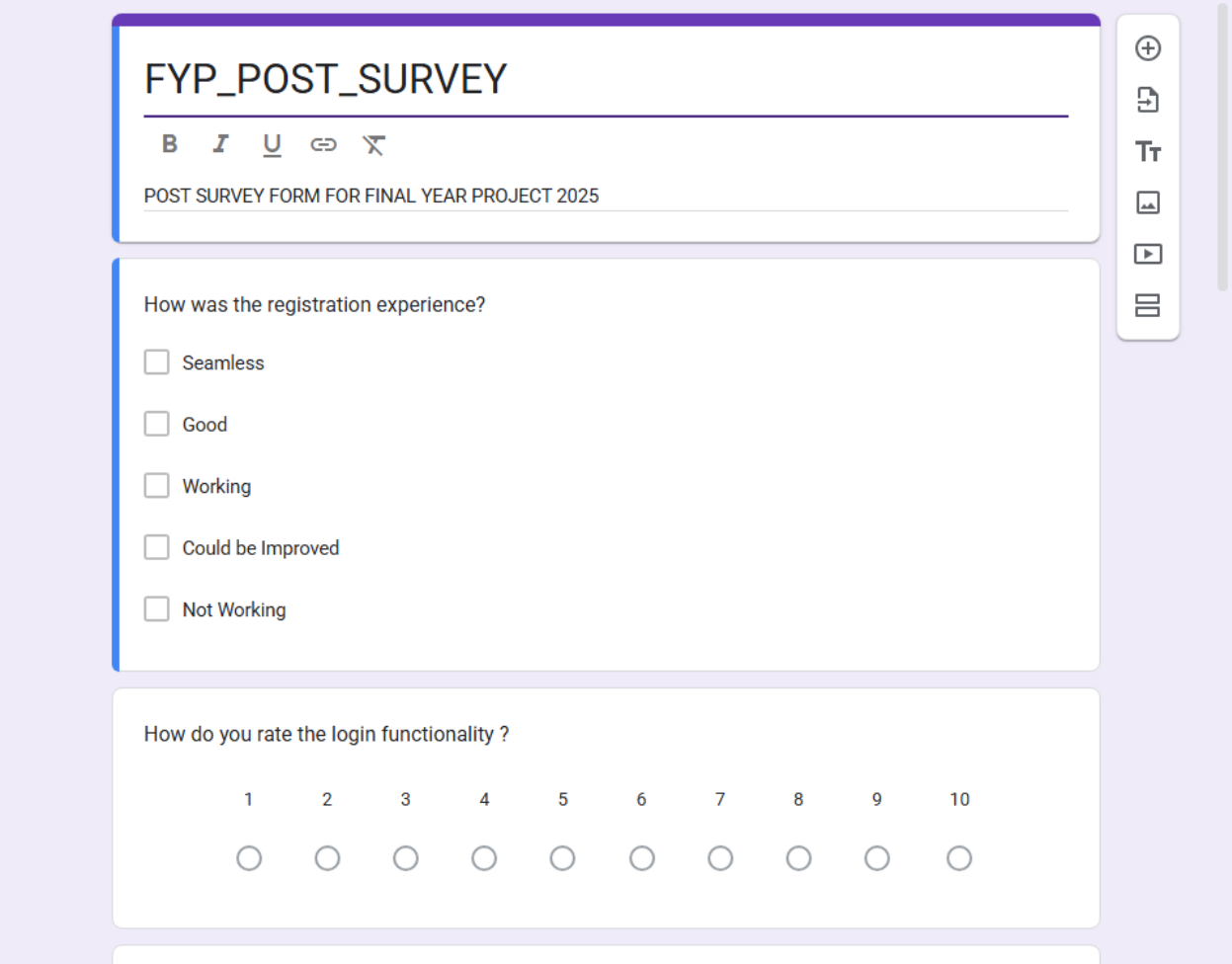


Figure 184: Post survey form-1

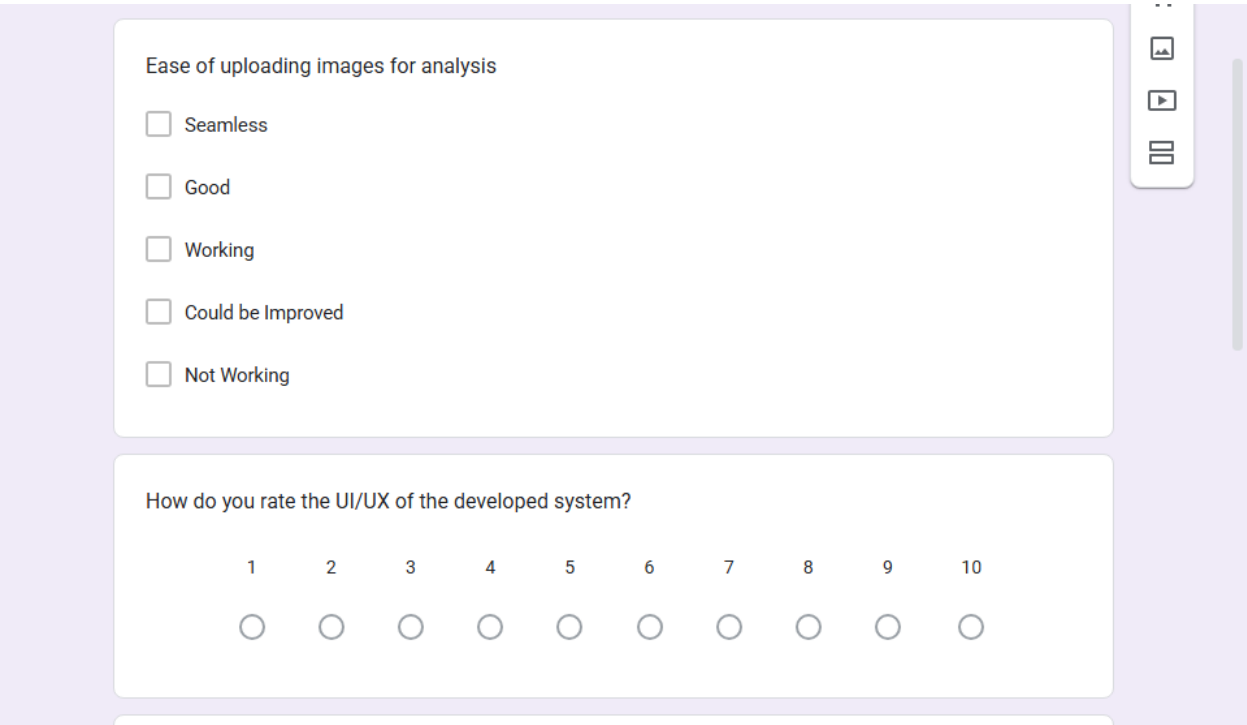


Figure 185: Post survey form-2

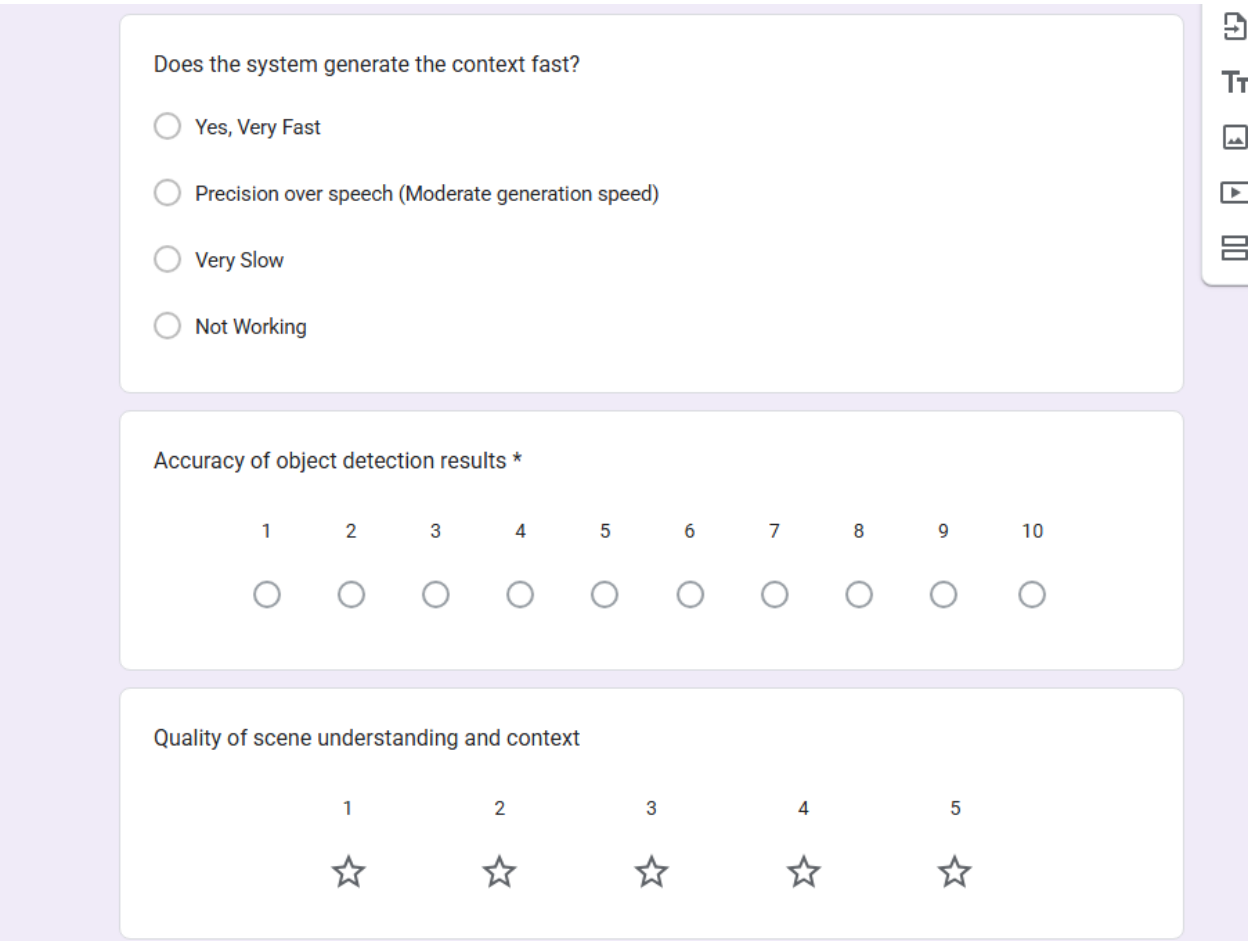


Figure 186: Post survey form-4

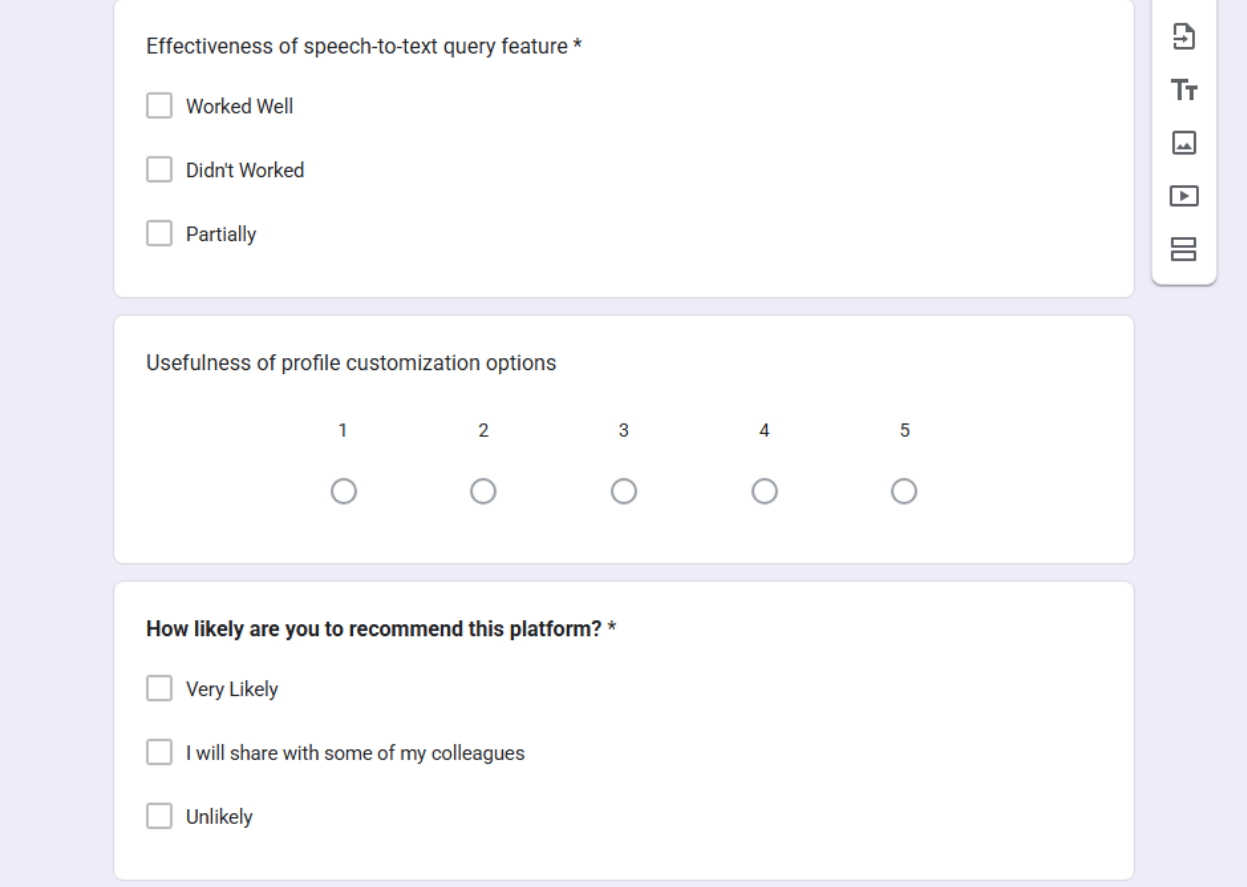


Figure 187: Post survey form-5

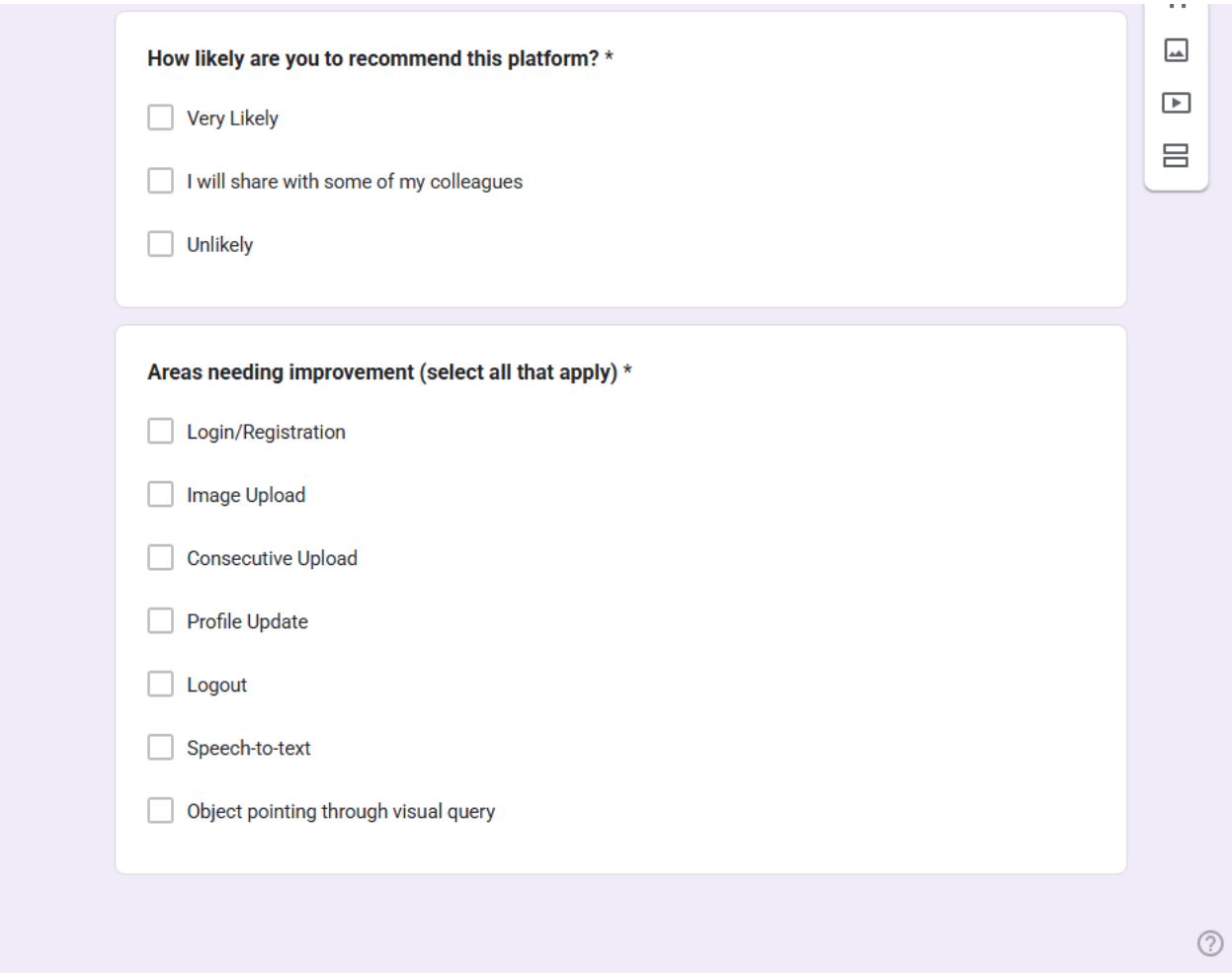


Figure 188: Post survey form-6

7.4. Appendix 4: Post Survey Results

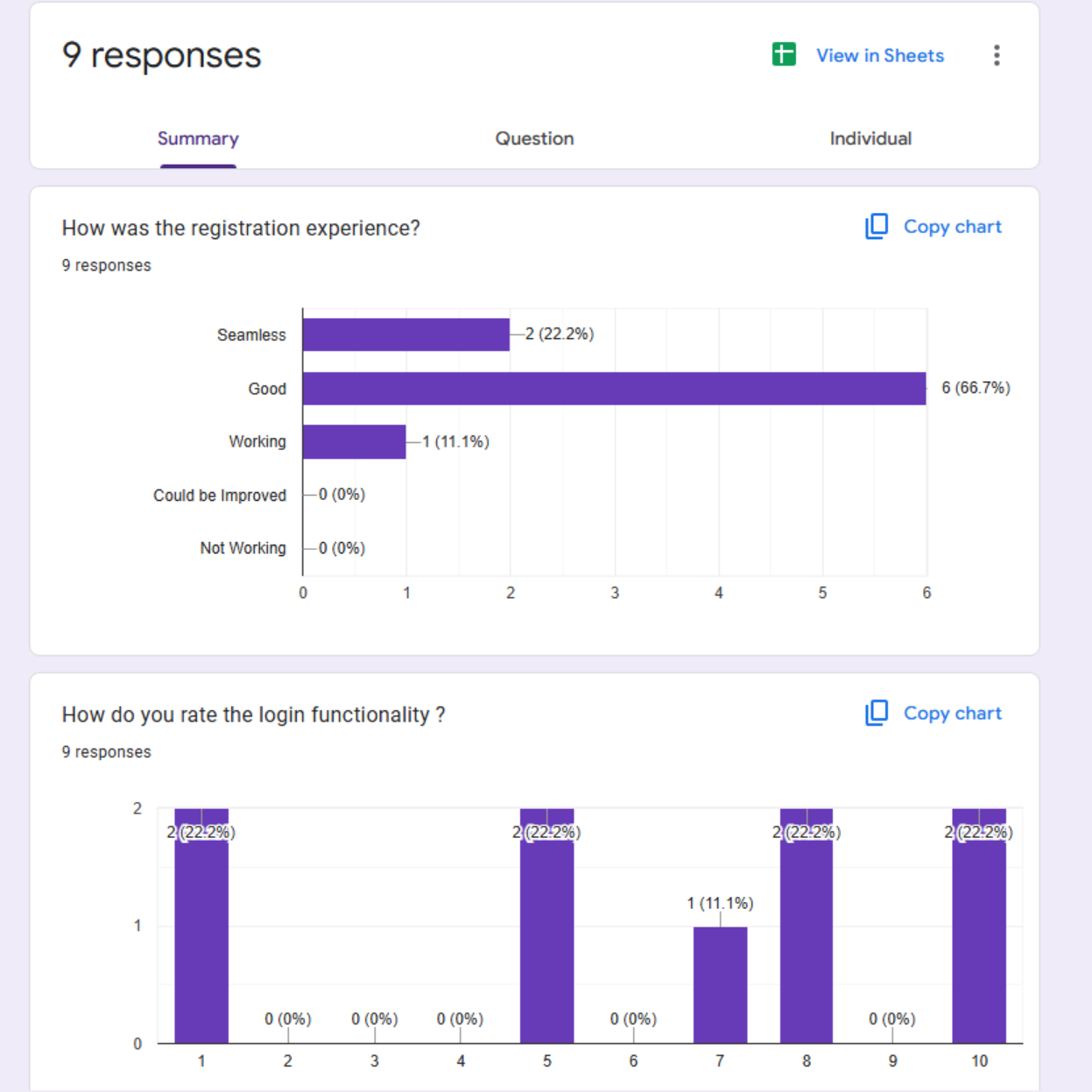


Figure 189: Post survey result-1

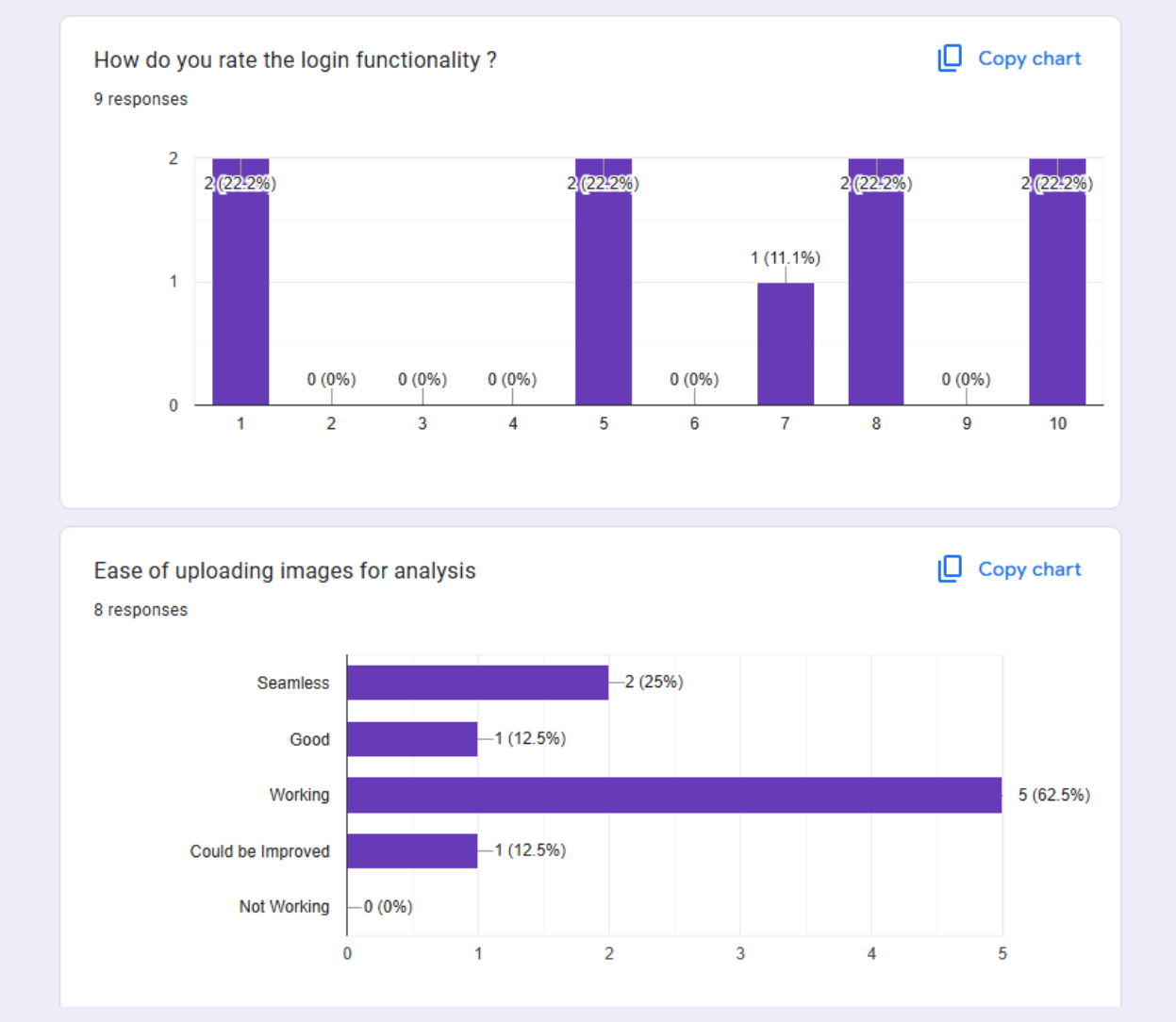


Figure 190:Post survey result-2

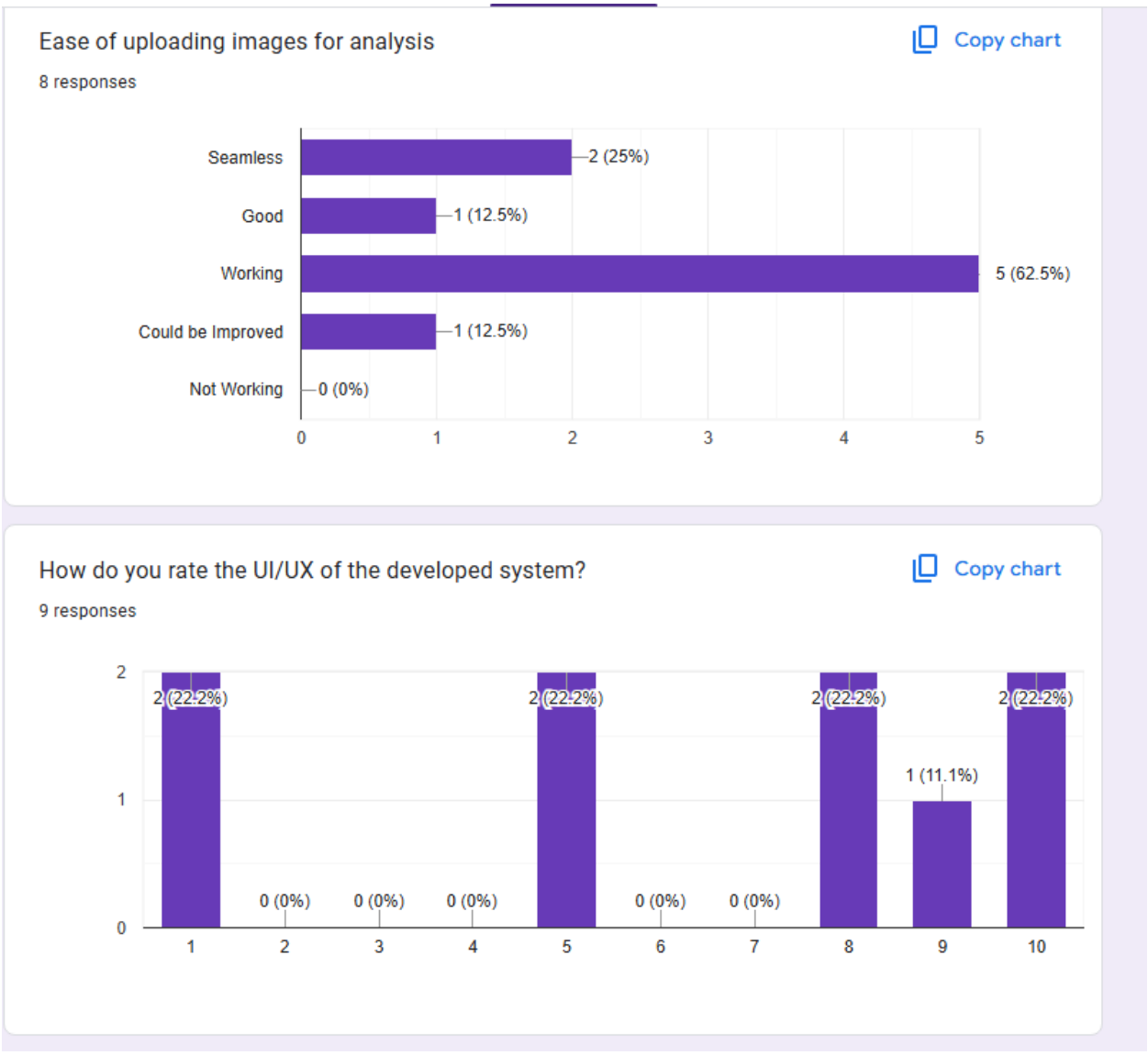


Figure 191: Post survey result-5

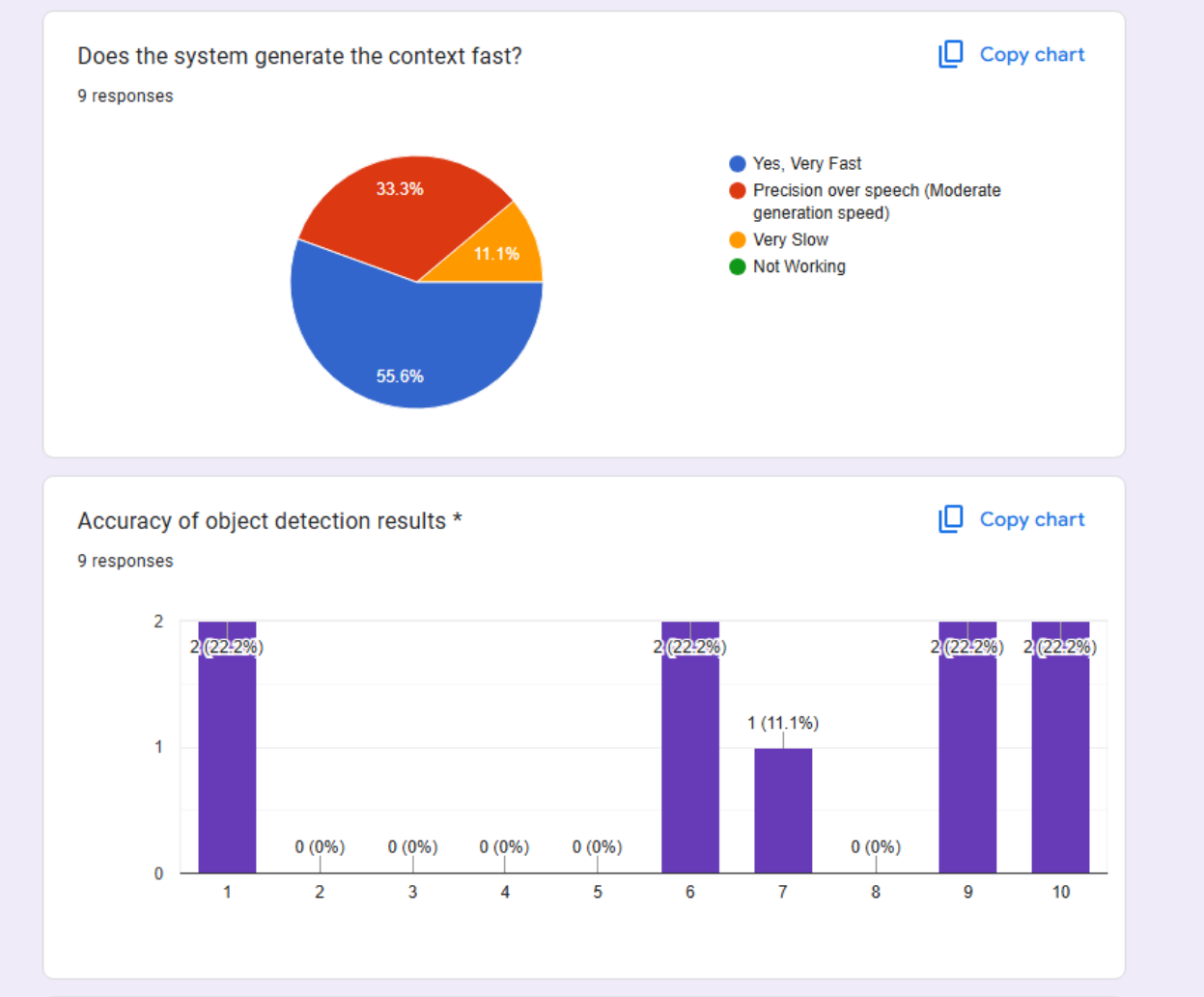


Figure 192: Post survey result-6

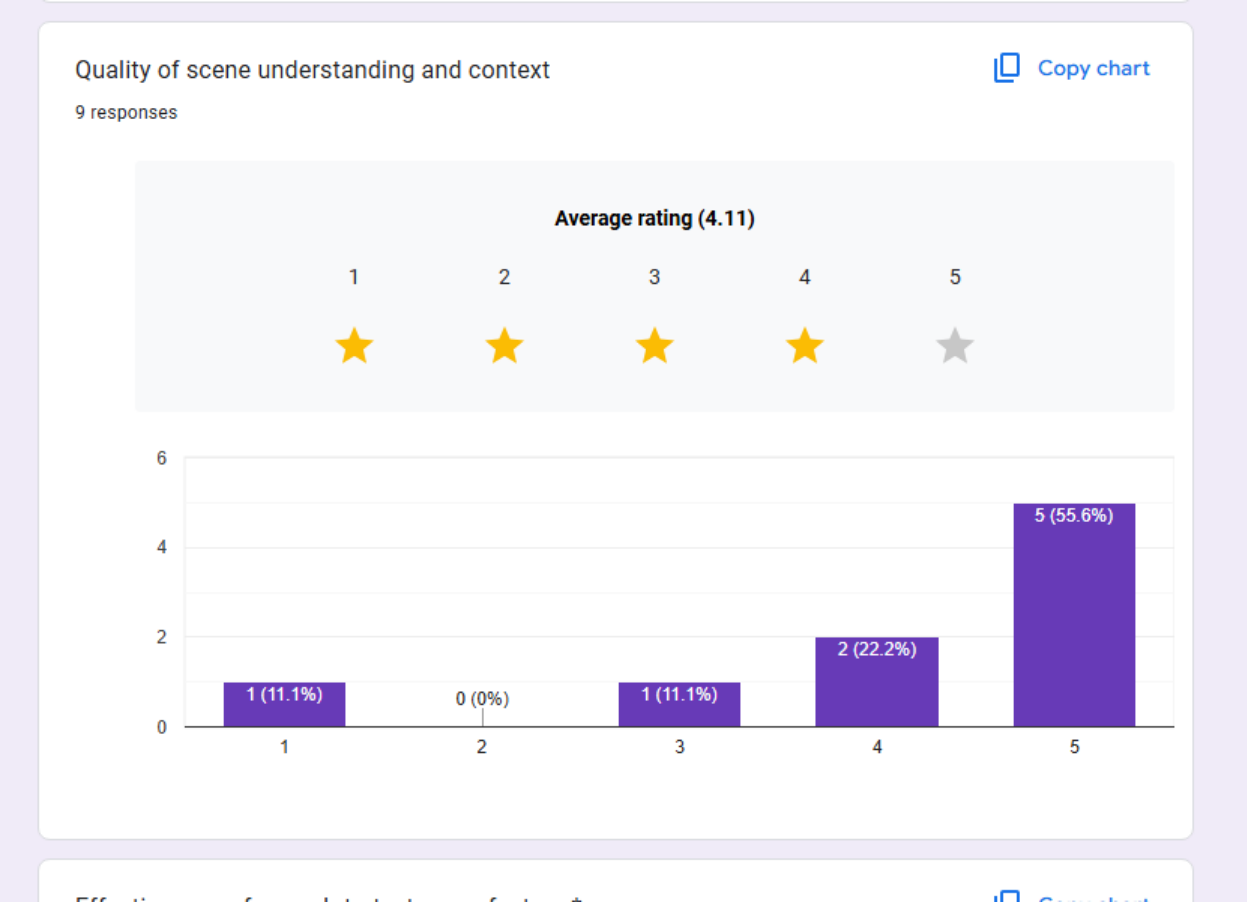


Figure 193: Post survey result-7

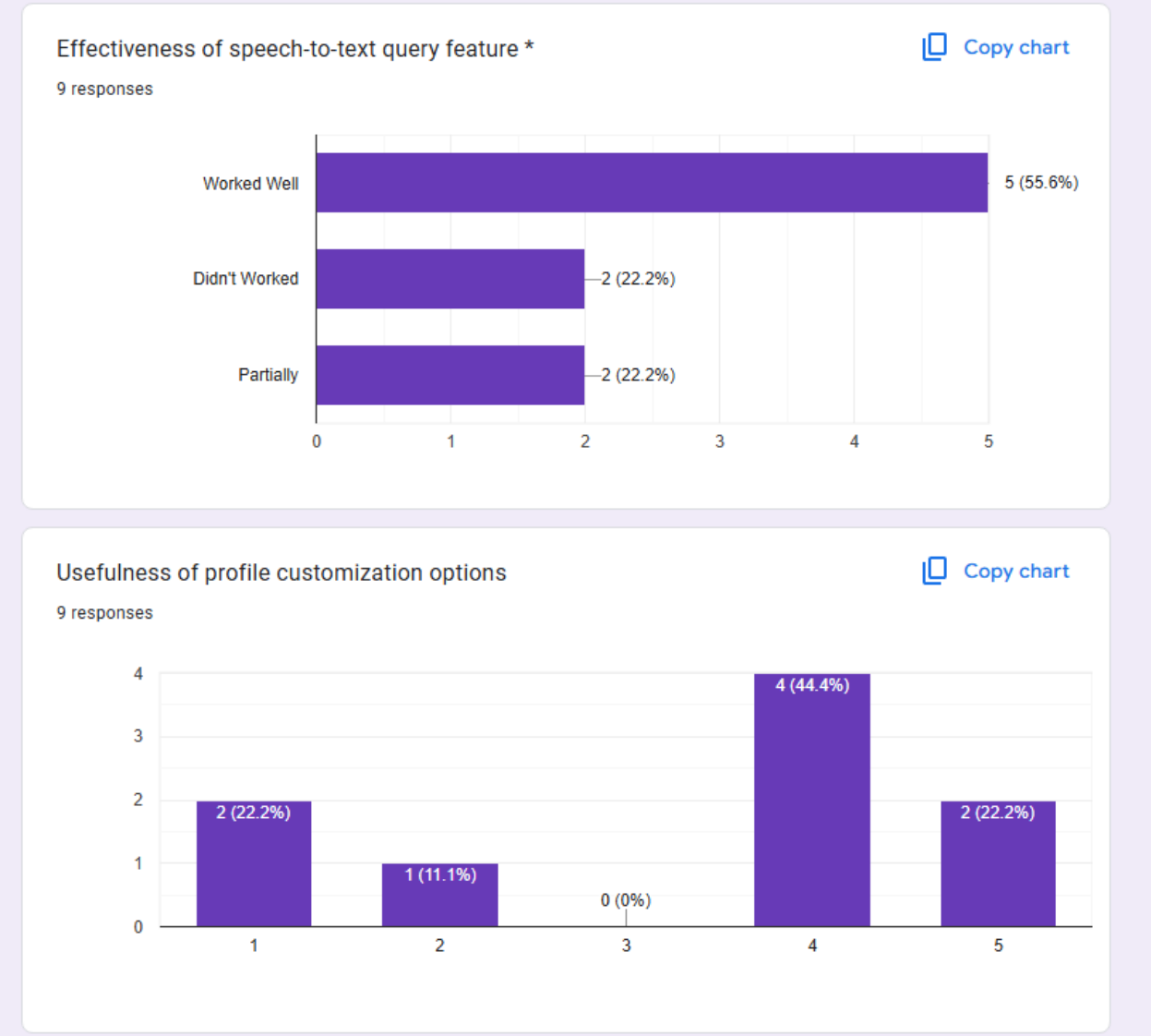


Figure 194: Post survey result-8

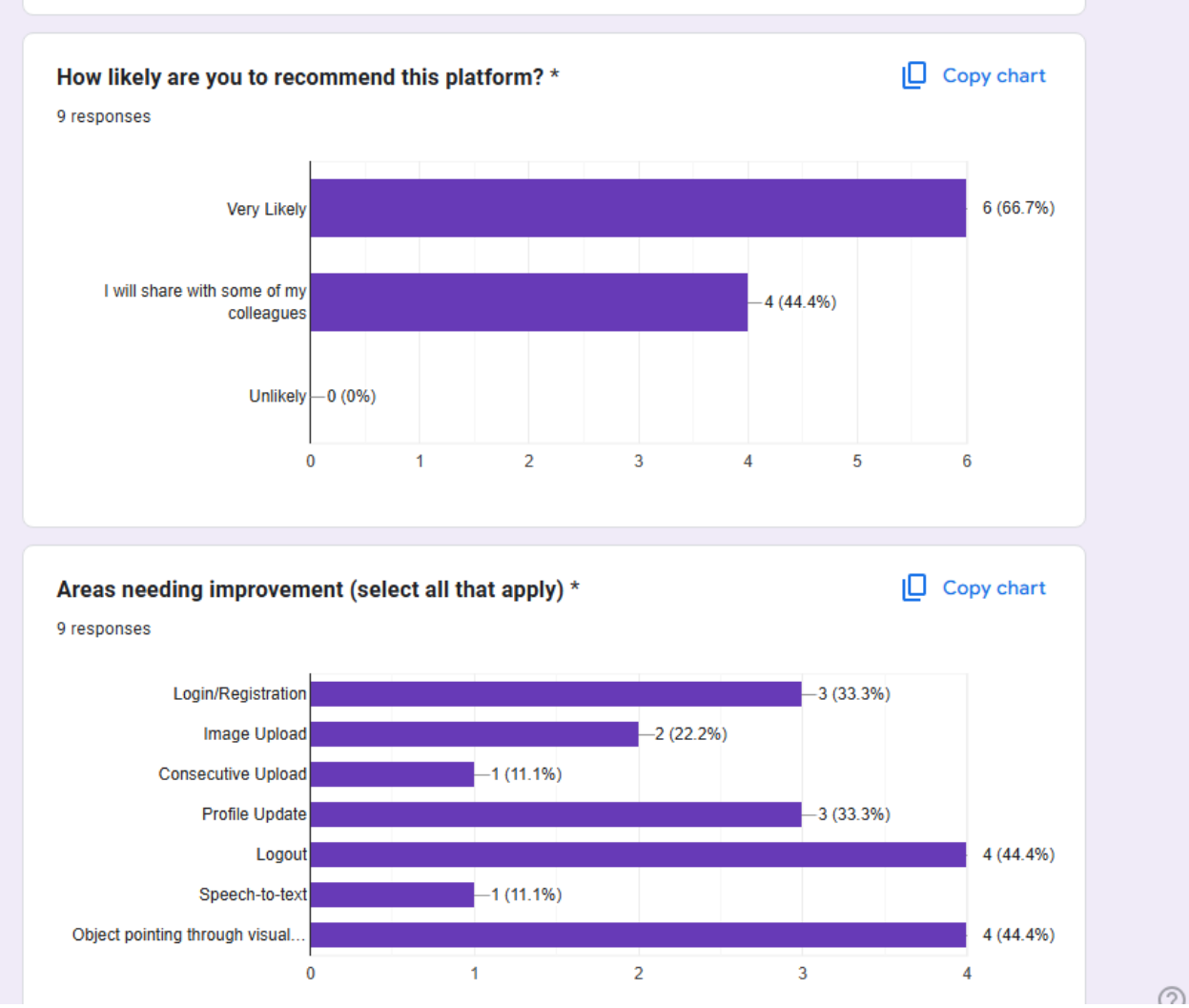


Figure 195: Post survey result- 9

7.5. Appendix **5: Important Features and Code Screenshots**

```python
    # Get optional query text
    query_text = request.POST.get('query_text', '')

    # Get user ID if user is authenticated
    user_id = request.user.id
    logger.info(f"Processing image for user: {request.user.username}")

    # Enqueue the job with required arguments
    try:
        queue = django_rq.get_queue('default')
        job = queue.enqueue(
            'blog.views.process_image_task',  # Use string reference to avoid import issues
            args=(image_file, query_text, user_id),  # Pass user_id as third argument
            job_timeout=1200,  # 20 minutes timeout (increased from 10)
            result_ttl=86400  # Results stored for 24 hours
        )

        # Return the job ID and initial response
        return JsonResponse({
            'job_id': job.id,
            'status': 'processing',
            'message': 'Image uploaded and processing started'
        })
    except Exception as redis_error:
        logger.error(f"Redis error: {str(redis_error)}")
        # Fall back to direct processing without Redis
        try:
            # Process directly without Redis
            analysis_id = process_image_task(image_file, query_text, user_id)
            if analysis_id:
                analysis = ImageAnalysis.objects.get(id=analysis_id)
                return JsonResponse({
                    'status': 'completed',
                    'message': 'Image processed directly (Redis unavailable)',
                    'image_url': analysis.image.url,
                    'short_caption': analysis.short_caption,
                    'query_text': analysis.query_text,
                    'query_result': analysis.query_result
                })
            else:
                return JsonResponse({
                    'status': 'failed',
                    'error': 'Failed to process image directly'
                }, status=500)
        except Exception as direct_error:
            logger.error(f"Direct processing error: {str(direct_error)}")
            return JsonResponse({
                'error': 'Error processing image (Redis and direct processing failed)'
```

Figure 196: Views Code-1

```python
    return redirect('blog:home')

if request.method == 'POST':
    form = UserRegistrationForm(request.POST)
    if form.is_valid():
        user = form.save()
        # Don't automatically log the user in
        messages.success(request, f'Account created successfully! Please log in.')
        # Redirect to login page instead of home
        return redirect('blog:login')
else:
    form = UserRegistrationForm()

return render(request, 'blog/register.html', {'form': form})

def login_view(request):
    """Handle user login."""
    if request.user.is_authenticated:
        return redirect('blog:home')

    next_url = request.GET.get('next', 'blog:home')

    if request.method == 'POST':
        form = UserLoginForm(request=request, data=request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(request=request, username=username, password=password)

            if user is not None:
                login(request, user)
                messages.success(request, f'Welcome back, {username}!')
                # Use the next parameter if available, otherwise go to home
                next_page = request.POST.get('next', next_url)
                return redirect(next_page)
        else:
            messages.error(request, 'Invalid username or password.')
    else:
        form = UserLoginForm()
```

Figure 197: Views Code-2

```python
    user_profile = request.user.profile

    if request.method == 'POST':
        form = UserProfileForm(request.POST, request.FILES, instance=user_profile)
        if form.is_valid():
            form.save()
            messages.success(request, 'Your profile has been updated!')
            return redirect('blog:profile')
    else:
        form = UserProfileForm(instance=user_profile)

    # Get user's analyses
    user_analyses = ImageAnalysis.objects.filter(user=request.user).order_by('-upload_date')[:5]

    context = {
        'form': form,
        'user_profile': user_profile,
        'user_analyses': user_analyses
    }

    return render(request, 'blog/profile.html', context)

login_required(login_url='blog:login')
ef admin_dashboard(request):
    """Render the admin dashboard with analytics."""
    try:
        # Log that we've reached the dashboard and authentication status
        logger.info(f"Admin dashboard accessed by user: {request.user.username}")
        logger.info(f"User authenticated: {request.user.is_authenticated}, Is staff: {request.user.is_staff}")

        # Proceed with dashboard data
        total_analyses = ImageAnalysis.objects.count()
        recent_analyses = ImageAnalysis.objects.select_related().order_by('-upload_date')[:5]

        # Calculate additional analytics data
        week_ago = timezone.now() - timedelta(days=7)
        recent_analyses_count = ImageAnalysis.objects.filter(upload_date__gte=week_ago).count()

        # Safely handle object counts - setting defaults
        total_objects = 0
        recent_objects_count = 0
```

Figure 198: Views Code-3

```python
    """
    import os
    from pathlib import Path

    # Build paths inside the project like this: BASE_DIR / 'subdir'.
    BASE_DIR = Path(__file__).resolve().parent.parent


    # Quick-start development settings - unsuitable for production
    # See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

    # SECURITY WARNING: keep the secret key used in production secret!
    SECRET_KEY = 'django-insecure-seagzefgq^%t@y+w&3*g*w6y2f+p67kfd8(k+w2758da(inbka'

    # SECURITY WARNING: don't run with debug turned on in production!
    DEBUG = True

    ALLOWED_HOSTS = []


    # Application definition

    INSTALLED_APPS = [
        'blog.apps.BlogConfig',
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
        'django_rq',
    ]

    MIDDLEWARE = [
        'django.middleware.security.SecurityMiddleware',
        'django.contrib.sessions.middleware.SessionMiddleware',
        'django.middleware.common.CommonMiddleware',
        'django.middleware.csrf.CsrfViewMiddleware',
        'django.contrib.auth.middleware.AuthenticationMiddleware',
        'django.contrib.messages.middleware.MessageMiddleware'
```

Figure 199: Settings Code

```python
    position.short_description = "Bounding Box"

@admin.register(ImageAnalysis)
class ImageAnalysisAdmin(admin.ModelAdmin):
    list_display = ['id', 'thumbnail', 'short_caption_preview', 'upload_date']
    list_display_links = ['id', 'thumbnail']
    search_fields = ['short_caption', 'query_text', 'query_result']
    list_filter = ['upload_date']
    readonly_fields = ['image_preview', 'upload_date', 'short_caption', 'query_text', 'query_result']
    fieldsets = [
        ('Image', {
            'fields': ['image', 'image_preview', 'upload_date']
        }),
        ('Generated Content', {
            'fields': ['short_caption'],
            'classes': ['wide']
        }),
        ('Visual Query', {
            'fields': ['query_text', 'query_result'],
            'classes': ['wide']
        })
    ]
    inlines = [DetectedObjectInline]

    def thumbnail(self, obj):
        if obj.image:
            return format_html('<img src="{}" width="50" height="50" style="object-fit: cover;" />', obj.image.url)
        return "No Image"

    thumbnail.short_description = "Image"

    def image_preview(self, obj):
        if obj.image:
            return format_html('<img src="{}" style="max-width: 500px; max-height: 500px;" />', obj.image.url)
        return "No Image"

    image_preview.short_description = "Image Preview"

    def short_caption_preview(self, obj):
        if obj.short_caption:
            # Truncate long captions for the list view
            max_length = 50
            return (obj.short_caption[:max_length] + '...') if len(obj.short_caption) > max_length else obj.short_caption
        return "No caption"

    short_caption_preview.short_description = "Caption"
```

Figure 200: Admin Code

```python
            self._use_cuda = torch.cuda.is_available()

            if self._use_cuda:
                # Get available GPU memory
                gpu_memory = torch.cuda.get_device_properties(0).total_memory / (1024**3)  # Convert to GB
                logger.info(f"GPU memory available: {gpu_memory:.2f} GB")

                # If GPU has less than 4GB, use CPU instead
                if gpu_memory < 4:
                    logger.warning(f"GPU memory ({gpu_memory:.2f} GB) is too low. Falling back to CPU.")
                    self._use_cuda = False

            device = torch.device("cuda" if self._use_cuda else "cpu")
            logger.info(f"Using device: {device}")

            # Initialize model with memory optimization
            if self._use_cuda:
                # For CUDA, use device_map="auto" with memory optimization
                self._model = AutoModelForCausalLM.from_pretrained(
                    "vikhyatk/moondream2",
                    revision="2025-04-14",
                    trust_remote_code=True,
                    device_map="auto",
                    torch_dtype=torch.float16,  # Use half precision to reduce memory usage
                    low_cpu_mem_usage=True
                )
            else:
                # For CPU, use memory optimization
                self._model = AutoModelForCausalLM.from_pretrained(
                    "vikhyatk/moondream2",
                    revision="2025-04-14",
                    trust_remote_code=True,
                    low_cpu_mem_usage=True
                )

            logger.info(f"Successfully loaded Moondream2 model on {device}")
```

Figure 201: Database Model Code

```python
import logging

# Configure logging
logger = logging.getLogger(__name__)

class SpeechRecognizer:
    def __init__(self, recording_time=15):
        # Set environment variables for CPU-only mode
        os.environ['CUDA_VISIBLE_DEVICES'] = ''  # Disable CUDA
        os.environ['USE_CUDA'] = '0'  # Force CPU mode

        try:
            # Initialize the recorder with CPU-only mode and minimal settings
            self.recorder = AudioToTextRecorder(
                model='tiny',  # Use tiny model for faster CPU processing
                language='en',
                device='cpu'
            )
            self.recording_time = recording_time
            self.transcribed_text = ""
            self.is_recording = False
            self._recording_thread = None
            self._stop_event = threading.Event()
            logger.info("Speech recognizer initialized successfully")
        except Exception as e:
            logger.error(f"Error initializing speech recognizer: {str(e)}")
            raise

    def start_recording(self, callback=None):
        """
        Start recording audio and convert to text.
        Stops after self.recording_time seconds.

        Args:
            callback: Function to call when recording is complete
        """
        if self.is_recording:
            return False
```

Figure 202: Speech to text Code

```
.gitignore  requirements.txt  model_handler.py  urls.py M
urls.py > ...
from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

app_name = 'blog'  # Namespace for main site URLs

# Main site URL patterns
urlpatterns = [
    # Public pages
    path('', views.home, name='home'),
    path('history/', views.history, name='history'),

    # Authentication
    path('register/', views.register, name='register'),
    path('login/', views.login_view, name='login'),
    path('logout/', views.logout_view, name='logout'),
    path('profile/', views.profile, name='profile'),

    # Image processing
    path('process-image/', views.process_image, name='process_image'),

    # Admin/Dashboard pages
    path('admin-dashboard/', views.admin_dashboard, name='admin_dashboard'),
    path('analyses/', views.image_analyses, name='image_analyses'),
    path('analysis/list/', views.analysis_list, name='analysis_list'),
    path('analysis/<int:pk>/', views.analysis_detail, name='analysis_detail'),
    path('analysis/<int:pk>/delete/', views.analysis_delete, name='analysis_delete'),
    path('check-job/<str:job_id>/', views.check_job_status, name='check_job_status'),

    # Analysis details
    path('analysis-details/<int:analysis_id>/', views.get_analysis_details, name='analysis_details'),

    # Speech to text
    path('speech-to-text/', views.speech_to_text, name='speech_to_text'),
]
```

Figure 203: URL Routing Codee

```python
self._use_cuda = torch.cuda.is_available()

if self._use_cuda:
    # Get available GPU memory
    gpu_memory = torch.cuda.get_device_properties(0).total_memory / (1024**3)  # Convert to GB
    logger.info(f"GPU memory available: {gpu_memory:.2f} GB")

    # If GPU has less than 4GB, use CPU instead
    if gpu_memory < 4:
        logger.warning(f"GPU memory ({gpu_memory:.2f} GB) is too low. Falling back to CPU.")
        self._use_cuda = False

device = torch.device("cuda" if self._use_cuda else "cpu")
logger.info(f"Using device: {device}")

# Initialize model with memory optimization
if self._use_cuda:
    # For CUDA, use device_map="auto" with memory optimization
    self._model = AutoModelForCausalLM.from_pretrained(
        "vikhyatk/moondream2",
        revision="2025-04-14",
        trust_remote_code=True,
        device_map="auto",
        torch_dtype=torch.float16,  # Use half precision to reduce memory usage
        low_cpu_mem_usage=True
    )
else:
    # For CPU, use memory optimization
    self._model = AutoModelForCausalLM.from_pretrained(
        "vikhyatk/moondream2",
        revision="2025-04-14",
        trust_remote_code=True,
        low_cpu_mem_usage=True
    )

logger.info(f"Successfully loaded Moondream2 model on {device}")
```

Figure 204: Context Generation Model Code

```python
def generate_short_caption(self, image):
    """Generate a short caption for the image."""
    try:
        if isinstance(image, str):
            image = Image.open(image)
        if image.mode != "RGB":
            image = image.convert("RGB")

        # Clear CUDA cache before processing if using CUDA
        if self._use_cuda:
            torch.cuda.empty_cache()

        result = self._model.caption(image, length="short")
        logger.info(f"Generated short caption: {result['caption']}")
        return result["caption"]
    except Exception as e:
        logger.error(f"Error generating short caption: {e}")
        raise

def generate_normal_caption(self, image):
    """Generate a normal caption for the image."""
    try:
        if isinstance(image, str):
            image = Image.open(image)
        if image.mode != "RGB":
            image = image.convert("RGB")

        # Clear CUDA cache before processing if using CUDA
        if self._use_cuda:
            torch.cuda.empty_cache()

        result = self._model.caption(image, length="normal", stream=True)
        caption = "".join(result["caption"])
        logger.info(f"Generated normal caption: {caption}")
        return caption
    except Exception as e:
        logger.error(f"Error generating normal caption: {e}")
        raise

def process_query(self, image, query="What is in this image?"):
    """Process a query about the image."""
    try:
        if isinstance(image, str):
            image = Image.open(image)
        if image.mode != "RGB":
            image = image.convert("RGB")
```

Figure 205: Context and Query Code

Go back to Implementation Sub-Section

7.6. Appendix 6: Agile Project Tracking Tool

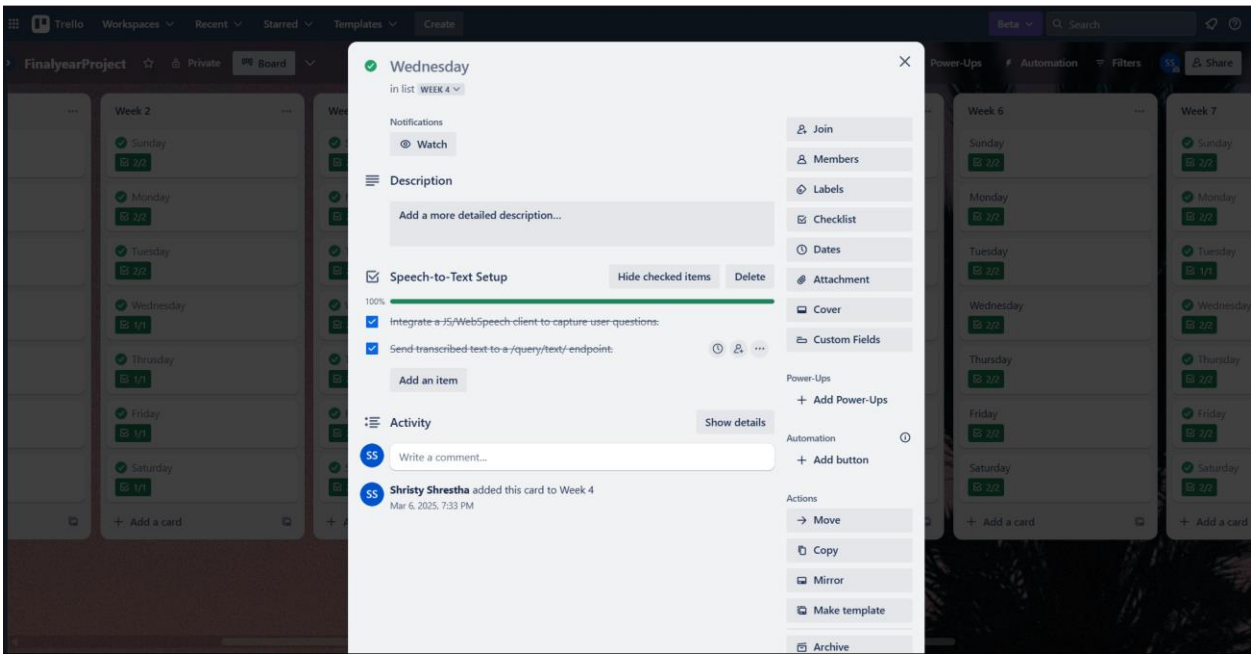


Figure 206: Trello Backlogs-1

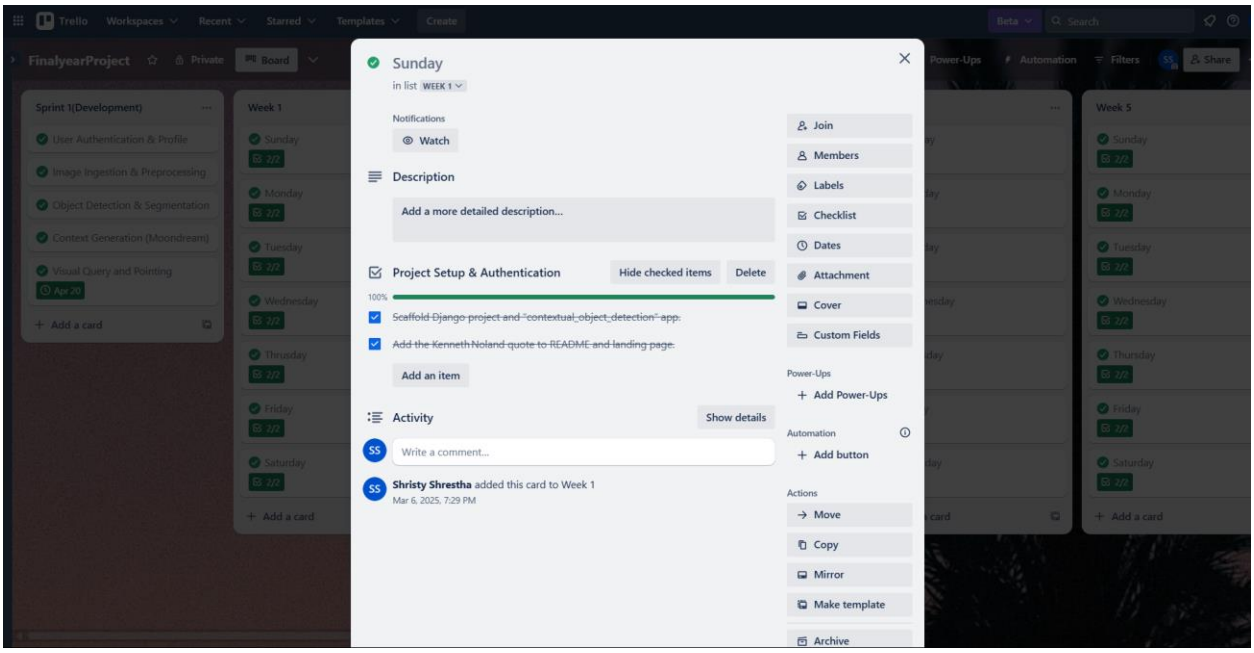


Figure 207: Trello Backlogs-2

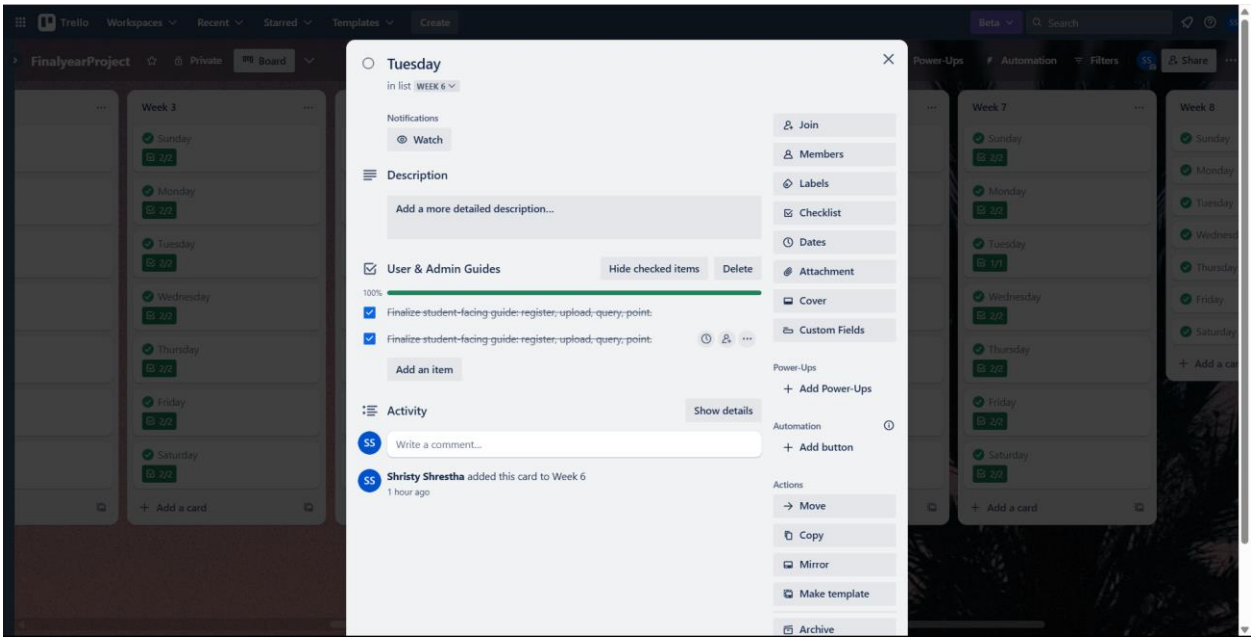


Figure 208: Trello Backlog-3

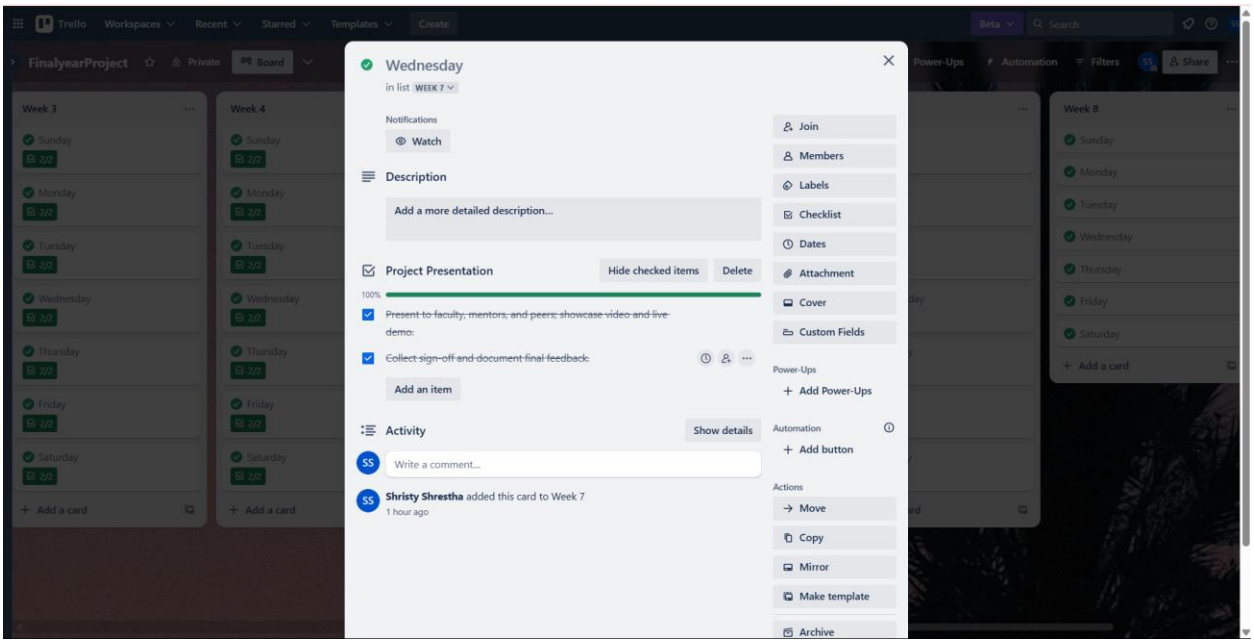


Figure 209: Trello Backlog-4

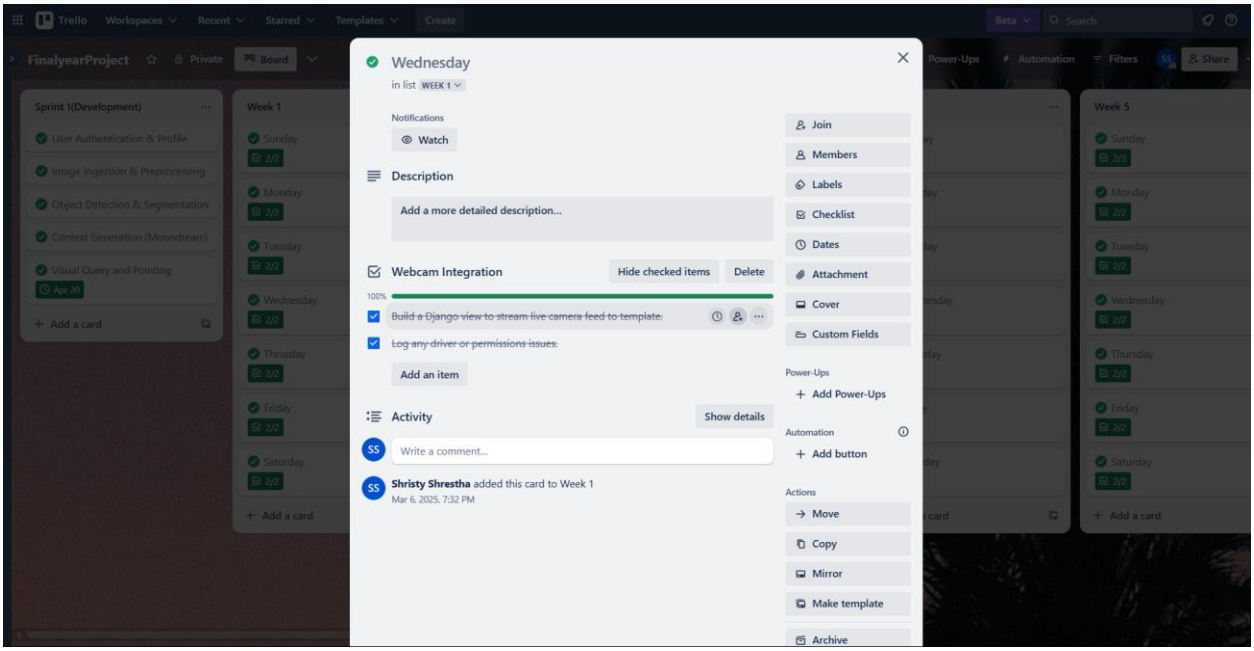


Figure 210: Trello Backlog-5

Go Back to Development Section

7.7. Appendix 7: Gantt Chart, Work Break Down Structure and Burndown Chart

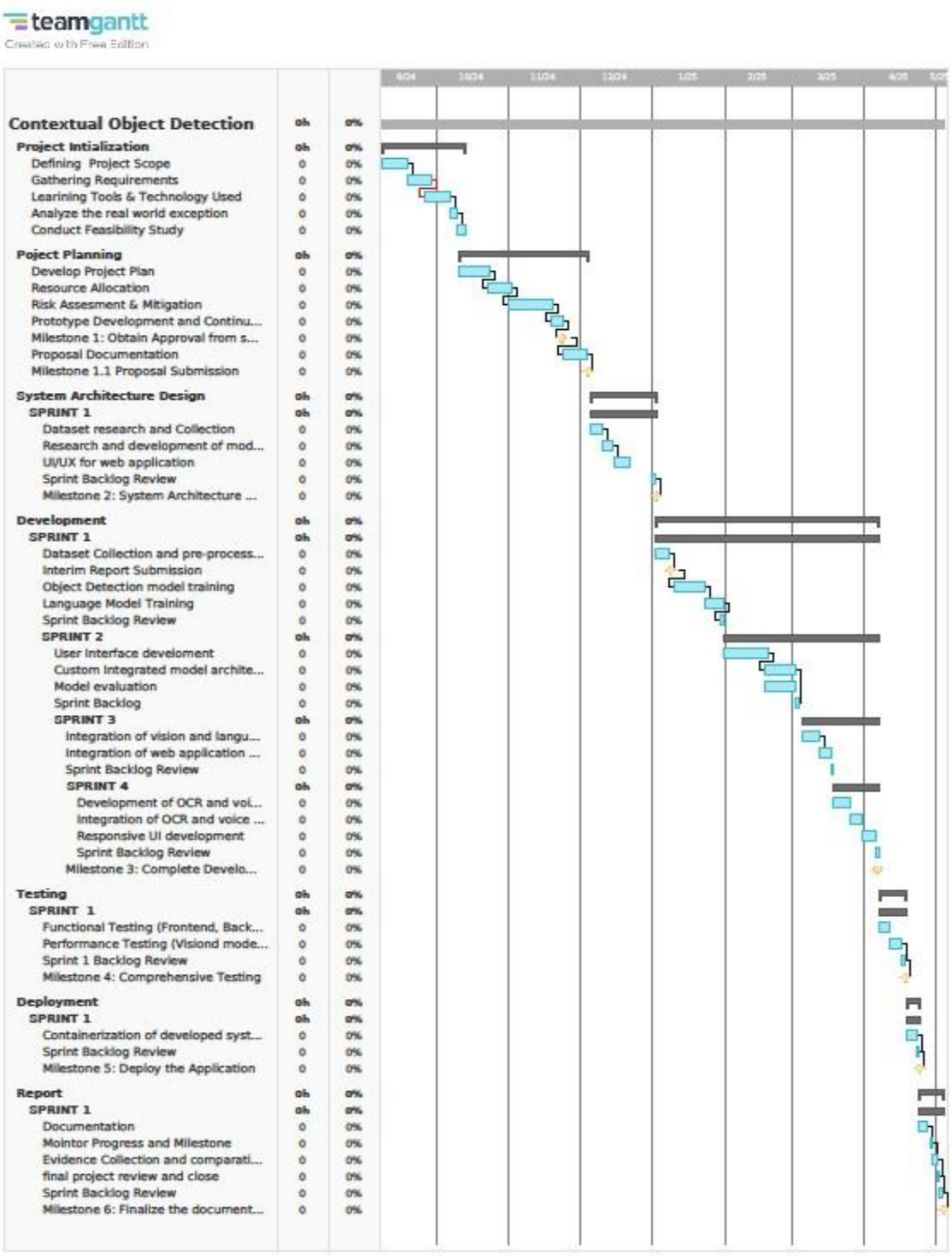


Figure 211: Gantt Chart for the project

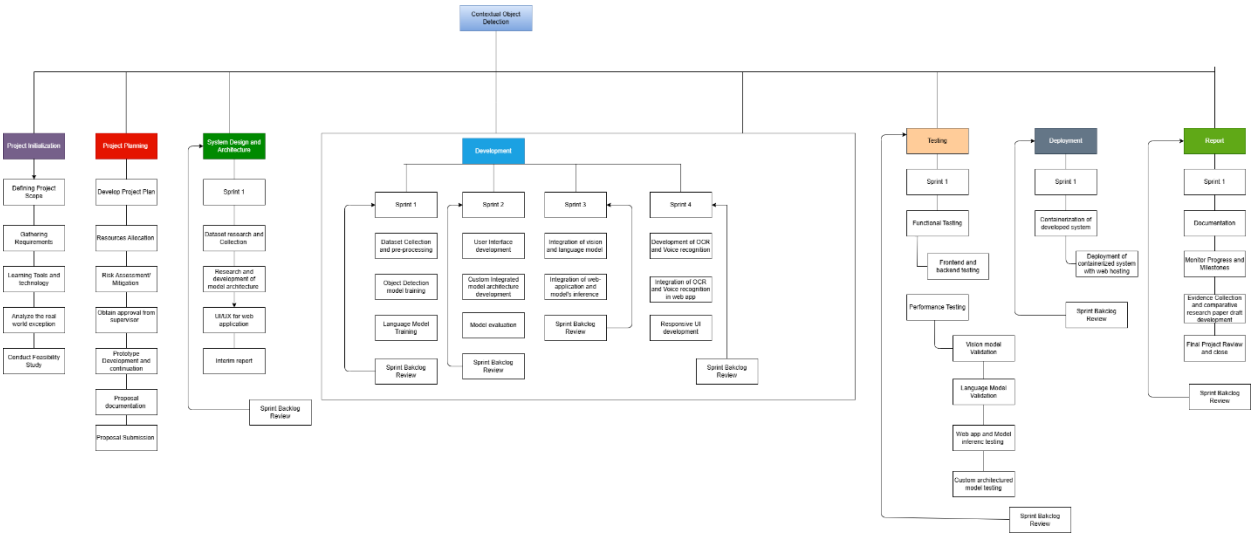


Figure 212: Work Breakdown Structure

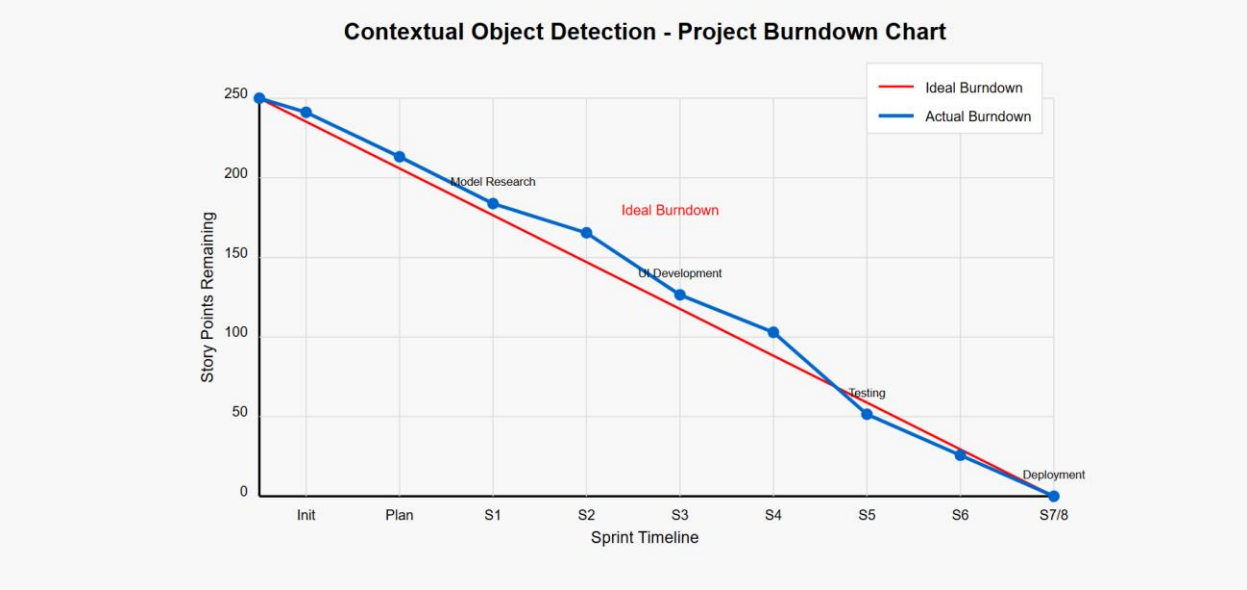


Figure 213: Project Burndown Chart

Back to Design Section