Dental AI Platform - Project Report

Project Title: Multi-Model AI System for Dental X-Ray Analysis **Author**: Harsha **Version**: 2.4 **Date**: December 2025

Executive Summary

This project developed a comprehensive dental AI platform that combines custom YOLOv8 object detection with multiple AI language models to provide accurate, visual analysis of dental X-rays. The system achieved 88% mAP accuracy in detecting dental pathologies and provides real-time analysis through an intuitive web interface.

---

1. Project Duration

Timeline: December 19 - December 22, 2025 **Total Duration**: Approximately 3-4 days of active development

Development Phases:

- **Day 1**: Initial concept, tech stack selection, and vision model integration
- **Day 2**: YOLO integration, multi-model text analysis, and custom training
- **Day 3**: Detection refinement, dataset quality improvements, and UI/UX fixes
- **Day 4**: Final optimizations, documentation, and code cleanup

---

2. Learning Outcomes

Technical Skills Acquired

1. **Computer Vision & Object Detection**

- Implemented YOLOv8 custom training pipeline
- Achieved 88% mAP@50 accuracy on dental X-ray dataset

- Developed class-specific confidence thresholds and spatial filtering
- Learned image preprocessing and augmentation techniques

2. **Multi-Model AI Integration**

- Integrated three AI models in parallel (GPT-4o-mini, Llama 3.3 70B, Qwen 3 32B)
- Implemented asynchronous API calls for optimal performance
- Built conversation context management system
- Designed message routing logic for different input types

3. **Web Application Development**

- Built production-ready Gradio interface with multiple tabs
- Implemented state management for conversation history
- Created interactive annotation playground
- Developed PDF report generation system

4. **Image Processing**

- Implemented CLAHE (Contrast Limited Adaptive Histogram Equalization) enhancement
- Developed proper grayscale-to-RGB conversion preserving full dynamic range
- Created bounding box annotation system with color coding
- Built image quality validation and filtering system

5. **Software Engineering Practices**

- Modular architecture with separation of concerns
- Error handling and resilience patterns
- Performance optimization (caching, async processing)
- Comprehensive documentation and code organization

Domain Knowledge Gained

- **Dental X-Ray Analysis**: Understanding of wisdom tooth impaction, dental pathologies, and X-ray interpretation
- **Medical AI Ethics**: Awareness of limitations and disclaimer requirements for medical AI systems
- **Dataset Management**: Experience with HuggingFace datasets, local dataset handling, and quality validation

---

3. Challenges Encountered

Challenge 1: Vision Model Hallucination

Problem: Initial implementation using GPT-4 Vision and Gemini Vision models produced inaccurate bounding box coordinates, often hallucinating detection locations.

Solution:

- Switched to YOLOv8 custom-trained model for accurate object detection
- Separated detection (YOLO) from analysis (text models)
- Implemented spatial filtering to reduce false positives

Impact: Detection accuracy improved from unreliable to 88% mAP@50.

Challenge 2: Image Corruption Issues

Problem: Images retrieved from HuggingFace dataset appeared corrupted (static noise, all black/white) in the Gradio interface.

Root Causes:

- Complex validation logic was corrupting images during processing
- Gradio PIL object serialization issues
- Dataset contained binary mask images mixed with real X-rays

Solution:

- Simplified image loading pipeline
- Switched to filepath-based display instead of PIL objects
- Implemented binary/mask image filtering (skips images with <10 unique values)
- Added proper grayscale-to-RGB conversion using NumPy stacking

Impact: All images now display correctly with proper grayscale preservation.

Challenge 3: Dataset Quality Issues

Problem: HuggingFace dataset contained many binary segmentation masks (only 2 unique pixel values) instead of real X-rays.

Solution:

- Implemented automatic filtering to skip binary images

- Added quality validation (unique values ≥10, mean brightness 5-250)
- Created retry mechanism (tries up to 50 indices to find valid X-rays)
- Built quality scanning tool to analyze dataset composition

Impact: System now only displays valid X-rays, improving user experience.

Challenge 4: Context Leakage in AI Models

Problem: AI models were inferring YOLO detection results from previous conversations even when no current image was present.

Solution:

- Modified context building to only include YOLO results when explicitly available
- Added explicit checks to prevent YOLO context inference
- Updated system prompts with critical instructions
- Implemented stricter None checks

Impact: Models now correctly state when X-ray analysis is needed.

Challenge 5: Performance Optimization

Problem: Sequential API calls were slow, and image loading was inefficient.

Solution:

- Implemented parallel async API calls for all three models
- Added LRU cache for dataset images (10-image cache)
- Optimized image processing pipeline
- Reduced detection inference time to <100ms

Impact: Overall response time improved significantly with parallel processing.

Challenge 6: YOLO Model Training and Hyperparameter Tuning

Problem: Initial YOLO model had high false positive rates, detecting impacted teeth in incorrect locations (middle of jaw instead of edges).

Root Causes:

- Pre-trained model was trained on generic objects, not dental X-rays
- Default confidence thresholds were too low

- No spatial filtering for anatomical constraints

Solution:

- Trained custom YOLO model on 1,075 dental X-rays from Roboflow dataset
- Implemented class-specific confidence thresholds (0.25 for Impacted, 0.30 for others)
- Added spatial filtering to restrict impacted tooth detections to jaw edges (x<0.30 or x>0.70)
- Fine-tuned IoU threshold (0.4) for better NMS (Non-Maximum Suppression)
- Reduced training time to 9 minutes using transfer learning

Impact: Detection accuracy improved from ~70% to 88% mAP@50, false positives reduced significantly.

Challenge 7: State Management Complexity

Problem: Managing conversation state, annotated images, and model selection across multiple tabs and interactions became complex and error-prone.

Issues Encountered:

- Annotated images disappearing during follow-up questions
- Conversation context not persisting correctly
- Model selection state not updating all displayed responses
- State synchronization between UI components

Solution:

- Implemented separate state variables: `conversation_state` (full context) and `stored_annotated_images` (persistent images)
- Created `update_model_selection()` function to update all historical responses when model changes
- Added explicit state management for image persistence across conversation turns
- Implemented proper state initialization and cleanup

Impact: Stable conversation flow with persistent images and consistent model switching.

Challenge 8: Gradio UI/UX Issues

Problem: Multiple UI/UX issues affected user experience, including image dragging in Firefox, modal display problems, and inconsistent button behaviors.

Specific Issues:

- Images in Annotation Playground were draggable in Firefox, interfering with click detection
- Fullscreen image modal not working correctly
- Button styling inconsistencies across different browsers
- Image gallery not displaying properly on mobile devices

Solution:

- Added CSS and JavaScript to prevent image dragging in Firefox
- Implemented proper modal overlay with ESC key support
- Created consistent button styling with hover effects
- Added responsive design considerations for mobile
- Fixed gallery display with proper image sizing constraints

Impact: Improved user experience across all browsers and devices.

Challenge 9: Dependency Management and Version Conflicts

Problem: Encountered version conflicts between different libraries, particularly with Gradio, PyTorch, and CUDA dependencies.

Issues Encountered:

- Gradio 6.x API changes breaking existing code
- PyTorch CUDA version mismatches
- httpx version conflicts with OpenAI API client
- PIL/Pillow version compatibility issues

Solution:

- Pinned specific versions in `requirements.txt` (e.g., `httpx<0.28`, `pillow==10.4.0`)
- Updated code to use Gradio 6.x API (`type="filepath"` instead of `type="pil"`)
- Created virtual environment isolation
- Documented exact dependency versions for reproducibility

Impact: Stable, reproducible environment across different systems.

Challenge 10: Error Handling and Edge Cases

Problem: System crashed or produced incorrect outputs when handling edge cases like empty images, API failures, malformed responses, or missing model files.

Edge Cases Encountered:

- API rate limiting causing failures
- Missing YOLO model file on first run
- Empty or corrupted image uploads
- Network timeouts during API calls
- Invalid bounding box coordinates

Solution:

- Implemented comprehensive try-catch blocks with detailed error messages
- Added fallback to base YOLOv8n model if custom model not found
- Created validation checks for image inputs
- Implemented retry logic for API calls
- Added graceful degradation (show error messages instead of crashing)
- Created user-friendly error messages with troubleshooting tips

Impact: Robust system that handles errors gracefully without crashing.

Challenge 12: Model Selection and Response Switching

Problem: When users switched between AI models (GPT-4o-mini, Llama, Qwen), only new responses showed the selected model, while historical responses remained unchanged, causing confusion.

Solution:

- Implemented `update_model_selection()` function that updates ALL historical responses
- Created mapping between conversation state entries and history display entries
- Preserved images and other metadata while updating response content
- Added visual feedback showing currently selected model

Impact: Consistent model selection across entire conversation history.

---

4. Use of AI Tools

AI Tools Used for Development

Primary Development Assistant: **Claude**

- Extensively used throughout the project for:
- Code generation and implementation
- Architecture design and refactoring
- Debugging and troubleshooting
- Documentation writing
- Code review and optimization suggestions

Key Interactions:

1. **Project Planning & Architecture** (Day 1)

- "How to structure the flow for a dental AI platform project"
- "Which tech stack is preferred given time constraints and deliverables"
- "Why is Gradio better than other frameworks for this use case?"
- Result: Comprehensive project structure and technology recommendations

2. **Initial Implementation** (Day 1)

- "Build a Gradio app with 2 tabs for dental AI platform"
- Detailed specifications for wisdom tooth detection and multi-model chatbot
- Result: Complete initial application structure with GPT-4 Vision and Gemini Vision integration

3. **YOLO Integration** (Day 2)

- "How to integrate YOLOv8 for accurate bounding box detection"
- "Implement custom YOLO training pipeline for dental X-rays"
- "Add class-specific confidence thresholds and spatial filtering"
- Result: Custom YOLO model integration replacing vision models

4. **Multi-Model AI Setup** (Day 2)

- "Implement parallel async API calls for GPT-4o-mini, Llama 3.3 70B, and Qwen 3 32B"
- "Create conversation context management system"
- "Handle model response formatting and display"
- Result: Robust multi-model system with parallel execution

5. **Image Processing Debugging** (Day 3)

- "Images from HuggingFace are corrupted - how to fix?"

- "Images are still corrupted (static/black) - debug step by step"
- "Implement proper grayscale-to-RGB conversion preserving full range"
- Result: Fixed image corruption through filepath-based display and proper conversion

6. **Dataset Quality Issues** (Day 3)
   - "HuggingFace dataset contains binary images - filter them out"
   - "Add automatic quality validation and retry mechanism"
   - "Create dataset quality scanning tool"
   - Result: Automatic filtering system skipping binary/mask images

7. **State Management** (Day 2-3)
   - "Fix annotated images disappearing during follow-up questions"
   - "Implement persistent image storage across conversation turns"
   - "Update all historical responses when model selection changes"
   - Result: Robust state management with persistent images

8. **Error Handling** (Throughout)
   - "Add comprehensive error handling for API failures"
   - "Implement fallback mechanisms for missing model files"
   - "Create user-friendly error messages with troubleshooting tips"
   - Result: Robust error handling throughout the application

9. **UI/UX Improvements** (Day 3)
   - "Fix image dragging issue in Firefox for Annotation Playground"
   - "Implement fullscreen image modal with ESC key support"
   - "Add consistent button styling and hover effects"
   - Result: Improved cross-browser compatibility and user experience

10. **PDF Report Generation** (Day 2-3)
    - "Create professional PDF report generation system"
    - "Include YOLO detections, AI analysis, and recommendations"
    - "Format clinical reports with proper styling"
    - Result: Complete PDF report generation feature

11. **Documentation** (Day 4)

- "Create consolidated documentation covering all aspects"
- "Write detailed architecture explanation with code examples"
- "Update all documentation to reflect current implementation"
- Result: Comprehensive documentation (1,200+ lines)

12. **Code Optimization** (Throughout)

- "Optimize async API calls for better performance"
- "Implement LRU cache for dataset images"
- "Reduce YOLO inference time"
- Result: Significant performance improvements

13. **Context Management Fix** (Day 2-3)

- "Fix AI models inferring YOLO results from previous conversations"
- "Prevent context leakage when no image is present"
- "Update system prompts with critical instructions"
- Result: Proper context isolation preventing false inferences

14. **Local Dataset Support** (Day 3)

- "Add automatic detection of local YOLO format datasets"
- "Implement fallback to HuggingFace if local dataset not found"
- "Create seamless switching between data sources"
- Result: Flexible dataset loading supporting both local and remote sources

15. **Code Cleanup** (Day 4)

- "Remove unnecessary test files and debug scripts"
- "Clean up Python cache files and temporary images"
- "Organize project structure"
- Result: Clean, production-ready codebase

Estimated Usage: 50+ prompts/interactions over the project duration

Interaction Patterns:

- **Problem-Solving**: Many interactions followed a pattern of describing a problem, getting AI suggestions, implementing, testing, and iterating
- **Code Generation**: AI generated initial code structures, which were then refined through multiple iterations

- **Debugging**: AI assisted in debugging complex issues through step-by-step analysis
- **Architecture Design**: AI provided architectural recommendations that were validated and adapted
- **Documentation**: AI helped structure and write comprehensive documentation

Example Interaction Flow:

Example 1: Image Corruption Debugging

```
User: "Images from HuggingFace are corrupted - how to fix?"
AI: [Suggests validation logic and image processing fixes]
User: "This is worse. Literally every image is static noise"
AI: [Suggests simplifying approach, removing complex validation]
User: "Images are still corrupted. Let's debug step by step"
AI: [Creates test scripts, identifies Gradio PIL serialization issue]
User: [Implements filepath-based display]
Result: Problem solved through iterative debugging
```

Example 2: YOLO Integration

```
User: "Vision models hallucinate coordinates. Need accurate detection"
AI: [Suggests YOLOv8 integration]
User: "How to train custom YOLO model for dental X-rays?"
AI: [Provides training pipeline with Roboflow dataset]
User: "Model has high false positives in wrong locations"
AI: [Suggests spatial filtering and class-specific thresholds]
Result: 88% mAP@50 accuracy achieved
```

Example 3: Multi-Model Setup

```
User: "Need to query 3 AI models in parallel"
AI: [Provides async implementation with asyncio]
User: "How to format and display responses from multiple models?"
AI: [Suggests response formatting and model selection UI]
User: "Historical responses don't update when switching models"
AI: [Provides update_model_selection() function]
```

```
  Result: Seamless multi-model experience
```

AI Assistance Effectiveness:

- **High Impact**: Architecture decisions, debugging complex issues, code structure
- **Medium Impact**: Implementation details, documentation writing, optimization
- **Low Impact**: Simple syntax questions, basic code snippets (could be done manually)

Time Saved: Estimated 60-70% reduction in development time through AI assistance, particularly in:

- Initial code scaffolding
- Debugging complex issues
- Writing comprehensive documentation
- Architecture design decisions

AI Models Integrated in the Application

1. **GPT-4o-mini (OpenAI)**

- Purpose: Text-based clinical analysis
- Usage: Parallel execution with other models
- Performance: Fast, accurate responses

2. **Llama 3.3 70B (via Groq)**

- Purpose: Alternative perspective on analysis
- Usage: Parallel execution for consensus
- Performance: High-quality medical reasoning

3. **Qwen 3 32B (via Groq)**

- Purpose: Third model for multi-model consensus
- Usage: Parallel execution
- Performance: Good balance of speed and accuracy

AI Tool Impact on Project

Positive Impacts:

- **Accelerated Development**: AI assistance significantly reduced development time

- **Code Quality**: AI suggestions improved code organization and best practices
- **Problem Solving**: AI helped debug complex issues (image corruption, context leakage)
- **Documentation**: AI assisted in creating comprehensive documentation

Challenges with AI Tools:

- **Hallucination**: Initial vision models produced inaccurate coordinates (led to YOLO solution)
- **Context Management**: Required careful prompt engineering to prevent context leakage
- **Iterative Refinement**: Multiple iterations needed for complex fixes (image corruption)

Lessons Learned:

- AI tools are powerful accelerators but require human oversight
- Domain-specific problems (medical imaging) benefit from specialized models (YOLO)
- Combining AI assistance with custom-trained models yields best results
- Critical to validate AI-generated code and solutions

---

5. Project Achievements

Technical Achievements

✅ **Custom YOLO Model**: Trained on 1,075 dental X-rays achieving 88% mAP@50 ✅ **Multi-Model System**: Successfully integrated 3 AI models with parallel execution ✅ **Real-Time Processing**: <100ms detection + parallel AI inference ✅ **Robust Image Handling**: Proper grayscale preservation and quality filtering ✅ **Production-Ready UI**: Full-featured Gradio interface with multiple tabs ✅ **PDF Report Generation**: Professional clinical report generation ✅ **Comprehensive Documentation**: 1,200+ lines of consolidated documentation

Key Metrics

- **Detection Accuracy**: 88% mAP@50
- **Inference Speed**: ~5ms per image (YOLO)
- **Training Time**: 9 minutes (GPU)
- **Response Time**: <3 seconds (including all 3 AI models)
- **Codebase Size**: ~3,500 lines of Python code
- **Documentation**: 1,200+ lines

---

6. Future Improvements

1. **Model Enhancements**
   - Expand YOLO training to more dental pathologies
   - Fine-tune confidence thresholds based on clinical feedback
   - Add support for panoramic X-rays
2. **User Experience**
   - Add user authentication and history saving
   - Implement batch processing for multiple X-rays
   - Add comparison view for before/after treatment
3. **Performance**
   - Implement model quantization for faster inference
   - Add GPU acceleration for YOLO detection
   - Optimize image caching strategy
4. **Clinical Integration**
   - Add DICOM file support
   - Integrate with dental practice management systems
   - Add HIPAA compliance features

---

7. Conclusion

This project successfully developed a production-ready dental AI platform that combines the accuracy of custom-trained YOLOv8 models with the reasoning capabilities of multiple AI language models. The system demonstrates the power of hybrid AI approaches, where specialized computer vision models handle detection while general-purpose language models provide clinical analysis.

The project provided valuable learning experiences in computer vision, multi-model AI integration, web development, and software engineering. The extensive use of AI development tools accelerated the development process while teaching important lessons about their capabilities and limitations.

Final Status: Production-ready system with comprehensive documentation and robust error handling.
