Aniket Patel

# Brag Sheet

## Experiences

### AI/ML Researcher at Ashland University

As an AI/ML Researcher at Ashland University, I developed a comprehensive end-to-end AI and IoT solution aimed at improving efficiency and decision-making in agriculture. The system combined a crop recommendation engine, an automated irrigation controller, and a plant disease detection model into a single integrated platform. Using environmental data such as temperature, humidity, and soil conditions, the crop recommendation model improved crop selection accuracy by 30 percent, while the AI-driven irrigation system dynamically adjusted water usage, reducing water waste by 25 percent. For plant health monitoring, I built and trained a deep learning-based disease detection model that achieved 98 percent classification accuracy on real-world crop images.

In parallel, I conducted extensive performance benchmarking of T5 Small, YOLOv5, and a custom CNN across CPU, GPU, and TPU environments to understand how different hardware platforms affect inference speed, memory usage, and scalability. This work helped identify optimal deployment strategies for large-scale AI workloads. I also evaluated multiple small and large transformer-based language models across 1,600 test cases to analyze bias, consistency, and reasoning reliability. Additionally, I compared regression models on datasets with varying feature complexity to guide model selection decisions, balancing interpretability, accuracy, and computational cost for real-world applications.

#### Technologies Used:

**Machine Learning**

- Supervised learning
- Regression models (linear, tree-based, ensemble)
- CNNs
- Transformer models
- T5 Small

**Computer Vision**

- YOLOv5

- Image classification
- Plant disease detection

**IoT & Systems**

- Sensor data ingestion
- Environmental data processing
- AI-driven automation logic

**Hardware & Acceleration**

- CPU inference
- GPU acceleration
- Google Cloud TPU
- Memory profiling

**Model Benchmarking**

- Performance evaluation
- Throughput and latency analysis
- Hardware-aware optimization

**Data Analysis**

- Bias detection
- Reasoning reliability evaluation
- Large-scale evaluation (1,600+ test cases)

**MLOps**

- Experiment tracking
- Model comparison frameworks
- Reproducibility pipelines

## Deep Learning Researcher at Wilkes University

During my research role at Wilkes University, I focused on protein-protein interaction site prediction, a critical task in computational biology and drug discovery. I designed and

implemented a custom deep learning model in TensorFlow that outperformed established architectures such as DeepPPISP and D-PPIsite by 15 percent in prediction accuracy. This improvement was achieved by closely analyzing the limitations of existing models and refining network architecture and feature representations to better capture interaction patterns in protein sequences.

To further enhance performance, I applied rigorous data preprocessing techniques, including normalization, outlier removal, and dimensionality reduction, which improved training stability and generalization. These optimizations led to an additional 25 percent boost in overall model performance, demonstrating the impact of high-quality data pipelines alongside architectural improvements. The research contributed to more reliable and efficient prediction of interaction sites, supporting downstream biological analysis and experimentation.

## Technologies Used:

**Deep Learning**

- TensorFlow
- Custom neural network architectures
- Sequence-based modeling

**Bioinformatics**

- Protein-protein interaction (PPI) prediction
- Biological sequence analysis

**Model Analysis**

- DeepPPISP
- D-PPIsite
- Architecture benchmarking

**Data Engineering**

- Data normalization
- Outlier detection and removal
- Dimensionality reduction (PCA, feature selection)

**Optimization**

- Hyperparameter tuning

- Training stability improvements
- Generalization enhancement

**Research Tools**

- Experimental evaluation pipelines
- Performance metrics analysis

# Projects

## Coursewiser

Coursewiser is a course-specific AI teaching assistant designed to help students better understand lecture material through accurate, context-aware responses. I fine-tuned the LLaMA 3.2 3B model using course lecture slides, notes, and instructor-provided materials, enabling the model to reason specifically within the context of a given class rather than relying on generic knowledge. The training followed a two-phase fine-tuning pipeline: the first phase aligned the model with course-specific semantics, while the second phase focused on instruction-response behavior to improve clarity, coherence, and reasoning quality. This approach increased academic query understanding accuracy by 70 percent and improved response quality by 45 percent. The optimized model was deployed on AWS EC2 as a low-latency API and integrated into a React-based web application using a Node.js backend and Firebase for authentication and data management. A feedback-driven refinement loop allowed continuous improvement based on student interactions, resulting in a highly responsive and practical learning assistant.

### Technologies Used:

**AI / Machine Learning**

- LLaMA 3.2 3B
- Transformer-based LLM fine-tuning
- Instruction tuning
- Context-aware reasoning models
- Supervised fine-tuning (SFT)
- Prompt alignment techniques

**Data & Training**

- Lecture slides preprocessing (PDF, PPT)

- Course notes and structured academic text
- Dataset curation and semantic alignment
- Feedback-driven refinement loops

**Backend & APIs**

- Node.js
- RESTful APIs
- Model inference APIs
- Low-latency request handling

**Frontend**

- React
- JavaScript / TypeScript
- Responsive UI design

**Cloud & Deployment**

- AWS EC2
- Model optimization for inference
- API deployment and scaling

**Auth & Storage**

- Firebase Authentication
- Firebase Firestore
- User interaction logging

**Dev Tools**

- Git
- Environment-based configuration
- Model versioning

Aniket Patel

# AI Theft Detection System

This project involved building a real-time, multimodal AI system for intelligent theft detection in surveillance environments. I integrated YOLOv8 for real-time object detection, VideoMAE for motion and temporal activity analysis, and a reasoning-capable multimodal LLM to interpret context across frames. By combining visual object cues with motion patterns and high-level reasoning, the system achieved a 65 percent improvement in detection accuracy compared to traditional rule-based systems. To reduce false positives, I designed an AI reasoning layer that cross-validated detected objects and actions over time, cutting false alerts by 40 percent. The system was implemented using OpenCV for video processing and deployed through a FastAPI backend that handled real-time inference, asynchronous processing, and event logging. A full reporting pipeline with SQLite storage and auto-generated HTML reports reduced manual video review time by 80 percent, significantly improving operational efficiency for security teams.

## Technologies Used:

**Computer Vision & Video**

- YOLOv8 (real-time object detection)
- VideoMAE (temporal motion understanding)
- OpenCV (video processing, frame extraction)

**Multimodal AI**

- Multimodal Large Language Models (vision + reasoning)
- Temporal context modeling
- Cross-modal reasoning pipelines

**Backend & Systems**

- FastAPI
- Asynchronous task processing
- Real-time inference pipelines

**Data Processing**

- Video stream handling
- Temporal windowing
- Event-based detection logic

**Databases & Reporting**

- SQLite
- Structured event logging
- Automated HTML report generation

**Optimization**

- False-positive reduction algorithms
- Temporal consistency checks
- Alert prioritization

**Deployment**

- Edge-ready inference design
- REST-based alert APIs

## Fathom

Fathom is an AI-powered desktop application that enables users to analyze databases using natural language instead of writing complex SQL queries. Built using Electron and React for the desktop interface and FastAPI for the backend, the application allows users to ask questions like “show monthly revenue trends” and receive accurate query results and visualizations instantly. I implemented a hybrid AI + SQL pipeline where a locally hosted LLM, served through Ollama, converts natural language into structured queries, enabling fully offline and privacy-preserving analytics. SQLAlchemy was used to safely execute generated queries, while Plotly was integrated to produce interactive, data-linked visualizations. This design reduced manual SQL querying time by 80 percent and significantly improved clarity and trust in results by tying AI interpretations directly to verifiable database outputs.

### Technologies Used:

**Desktop & Frontend**

- Electron
- React
- TailwindCSS
- TypeScript

**Backend**

- FastAPI
- REST APIs
- Async request handling

**AI & NLP**

- Locally hosted LLMs
- Ollama
- Natural language to SQL translation
- Semantic parsing

**Databases & Querying**

- SQLAlchemy
- Relational databases (PostgreSQL / SQLite compatible)
- Query validation and execution

**Data Visualization**

- Plotly
- Interactive charts and dashboards
- Data-linked visual analytics

**Privacy & Offline**

- Fully offline execution
- Local inference pipelines
- Privacy-preserving analytics

**Dev & Packaging**

- Desktop app packaging
- Cross-platform builds
- API-to-UI data binding

Aniket Patel

# MarketMinds

MarketMinds is a multi-agent AI system designed to automate end-to-end product and market research. Using CrewAI and the Gemini LLM, I built a coordinated group of specialized agents, including Product, Competitor, Review, and Strategist agents, each responsible for a specific research task. LangChain was used to orchestrate agent workflows, manage shared context, and enable reasoning across agents. The system pulls real-time data using SerpAPI and processes it through a FastAPI backend, while a React frontend presents actionable insights and strategic recommendations. By automating data collection, analysis, and synthesis, MarketMinds reduced manual research time by 60 percent and provided faster, more consistent market intelligence for product teams.

## Technologies Used:

**AI & Agents**

- CrewAI
- Multi-agent LLM systems
- Role-based agent reasoning
- Autonomous task planning

**LLMs**

- Gemini LLM
- Prompt engineering
- Chain-of-thought orchestration

**Orchestration**

- LangChain
- Agent pipelines
- Tool-calling frameworks

**Data & Search**

- SerpAPI
- Real-time web data ingestion

**Backend**

- FastAPI
- API aggregation services
- Data normalization pipelines

**Frontend**

- React
- Dynamic dashboards
- Insight visualization

**System Design**

- Agent-to-agent communication
- Modular pipeline architecture
- Scalable research workflows

## Prepwiser

Prepwiser is an AI-powered mock interview platform designed to help candidates prepare more effectively for technical and behavioral interviews. I built the frontend using React and integrated Supabase for authentication and data storage, while LiveKit handled real-time audio and video communication. Tavus was used to create realistic AI interviewers that simulate real interview conditions. The platform generates personalized interview questions by analyzing a user's resume and job description, increasing relevance and personalization by 60 percent. After each session, the system provides instant AI-driven feedback that highlights strengths, weaknesses, and concrete next steps, leading to a 40 percent improvement in interview performance. Overall, the platform doubled candidate preparedness by combining realism, personalization, and immediate, actionable feedback.

### Technologies Used:

**Frontend**

- React
- Responsive UI
- Session-based workflows

**Backend & Services**

- Supabase (Auth, DB, Storage)

- REST APIs

**Real-Time Communication**

- LiveKit (video/audio streaming)
- WebRTC-based conferencing

**AI & Personalization**

- Resume parsing
- Job description analysis
- Personalized question generation
- Feedback generation models

**Video AI**

- Tavus (AI interviewers)
- Realistic avatar-driven interviews

**Analytics**

- Performance scoring systems
- Strength/weakness detection
- Feedback pipelines

## CodDoc

CodDoc is an AI-powered developer productivity tool that automatically generates professional, well-structured README files for GitHub repositories by analyzing the actual codebase instead of relying on templates or keywords. The system uses a multi-agent architecture where five specialized AI agents collaboratively inspect the repository structure, source code files, dependencies, and configuration files such as package.json to understand the project's purpose, setup, and usage. This agent-based reasoning approach allows CodDoc to produce accurate, context-aware documentation that follows industry best practices.

I built CodDoc with a modern, responsive frontend that provides a smooth user experience, allowing developers to simply paste a GitHub repository URL and receive a fully formatted README within seconds. The AI pipeline is powered by Gemini AI and orchestrated using LangGraph to coordinate agent workflows and reasoning steps. The generated documentation can be edited directly within the app and exported for immediate use, reducing documentation

effort from hours to seconds. CodDoc has been deployed to production with multiple releases and deployments, demonstrating stability and real-world usage, and significantly improving documentation quality and developer onboarding efficiency.

## Technologies Used:

**Frontend**

- Next.js 14
- React
- TypeScript
- Tailwind CSS
- Framer Motion
- shadcn/ui

**AI & Agent Systems**

- Gemini AI
- LangGraph
- Multi-agent LLM architecture
- Codebase and dependency analysis

**Backend & Services**

- API-based AI inference
- Environment-based configuration (.env)

**DevOps & Deployment**

- Production deployments (Vercel-style CI/CD)
- Environment variable management

**Developer Tools**

- Git
- GitHub
- Node.js (18+)
- npm

Aniket Patel

# Technical Skills

## Programming Languages:

Python, Java, C++, SQL (MySQL, PostgreSQL), JavaScript, TypeScript, Bash

## AI/ML & Libraries:

PyTorch, TensorFlow, Transformers, CNNs, Scikit-Learn, Pandas, NumPy, SciPy, Matplotlib, Seaborn, Plotly, LangChain, LangGraph, CrewAI

## Computer Vision & NLP:

YOLOv5, YOLOv8, VideoMAE, ResNet, LLaMA, BERT, GPT-style Models, T5, RAG, NLTK

## MLOps & Deployment:

FastAPI, Flask, Docker, Kubernetes, ONNX, TensorRT, MLflow, RESTful APIs

## Cloud & Tools:

AWS, GCP, Microsoft Azure, Git, GitHub, Jupyter Notebook, Google Colab, Kaggle, Visual Studio