Agentic AI Certification Training Course

Capstone Project

Capstone Project: Intelligent User Feedback Analysis and Action System

Business Problem

Modern SaaS and app-based companies receive **dozens of user reviews and feedback daily** from multiple channels including app stores (Google Play, App Store), customer support emails, and user surveys. The current **manual triaging process** is slow, inconsistent, and doesn't scale effectively resulting in critical bugs being missed, feature requests being delayed, and inconsistent prioritization across teams.

Scenario

You are a product engineer at a B2C mobile app company that manages a productivity app with around 10,000 active users. Your team currently receives:

- 10-20 app store reviews daily
- 5-10 customer support emails per day
- Occasional in-app feedback submissions

A team member manually reads through this feedback and creates tickets in your project management system. This process takes 1-2 hours daily and often results in:

- Delayed response to critical bugs
- Inconsistent ticket formatting and prioritization
- Lost or overlooked feedback
- Poor traceability from user complaint to engineering resolution

Your Task: Complete Project Implementation

Design, implement, and demonstrate a **complete multi-agent AI system** that:

1. **Reads** user feedback from CSV files containing app store reviews and support emails
2. **Classifies** content into categories (Bug / Feature Request / Praise / Complaint / Spam)

3. **Extracts** actionable insights and technical details
4. **Creates** structured tickets and logs them to CSV files with appropriate priority levels and metadata
5. **Ensures** quality and consistency through automated review
6. **Provides** a user interface for monitoring and manual overrides

System Objectives

Your complete system should achieve:

- **Automation**: Process feedback from CSV files without manual intervention
- **Speed**: Complete analysis and ticket creation within minutes
- **Consistency**: Standardize ticket format and priority assignment
- **Traceability**: Maintain clear links from original feedback to generated tickets
- **Usability**: Provide an intuitive interface for monitoring and control

Multi-Agent Architecture to Implement

Agent	Primary Responsibilities
CSV Reader Agent	Reads and parses feedback data from CSV files
Feedback Classifier Agent	Categorizes feedback using NLP (bug, feature request, praise, complaint, spam)
Bug Analysis Agent	Extracts technical details: steps to reproduce, platform info, severity assessment
Feature Extractor Agent	Identifies feature requests and estimates user impact/demand
Ticket Creator Agent	Generates structured tickets and logs them to output CSV files
Quality Critic Agent	Reviews generated tickets for completeness and accuracy

Technical Implementation Requirements

- **Framework**: CrewAI or AutoGen for agent orchestration
- **UI**: Streamlit for monitoring and manual overrides
- **Input**: Read from CSV files containing mock feedback data
- **Output**: Log generated tickets to CSV files for offline analysis
- **Error Handling**: Robust error handling and logging
- **Configuration**: Configurable parameters for classification thresholds and priorities

Step 1: Create Mock Dataset CSV Files

First, create the following CSV files with realistic sample data:

1. app_store_reviews.csv

Columns: review_id, platform, rating, review_text, user_name, date, app_version

Hints:

- **Bug examples**: "App crashes when I...", "Can't login since update", "Data sync not working"
- **Feature requests**: "Please add...", "Would love to see...", "Missing functionality..."
- **Praise**: "Amazing app!", "Love the new feature", "Works perfectly"
- **Complaints**: "Too expensive", "Poor customer service", "App is slow"
- **Spam**: Promotional text, random characters, unrelated content
- **Mix platforms**: Include both "Google Play" and "App Store"
- **Vary ratings**: 1-5 stars, with bugs typically 1-2, features 3-4, praise 4-5
- **Realistic versions**: "2.1.3", "3.0.1", etc.

2. support_emails.csv

Columns: email_id, subject, body, sender_email, timestamp, priority

Hints:

- **Bug subjects**: "App Crash Report", "Login Issue", "Data Loss Problem"
- **Feature subjects**: "Feature Request: Dark Mode", "Suggestion for Improvement"
- **Include technical details**: Device models, OS versions, steps to reproduce
- **Vary email styles**: Formal business emails vs casual user messages
- **Include timestamps**: Recent dates in various formats
- **Priority levels**: Leave some blank, others with "High", "Medium", "Low"

3. expected_classifications.csv

Columns: source_id, source_type, category, priority, technical_details, suggested_title

Hints:

- **Map to your reviews/emails**: Use same IDs from above files
- **Categories**: Bug, Feature Request, Praise, Complaint, Spam
- **Priorities**: Critical, High, Medium, Low
- **Technical details**: For bugs, include device info, reproduction steps
- **Suggested titles**: Clear, actionable ticket titles

Step 2: System Implementation

Implement the complete multi-agent system including:

Core Components:

1. **Agent Classes**: Implement each agent with specific responsibilities
2. **Data Processing Pipeline**: CSV reading, processing, and output generation
3. **Classification Logic**: NLP-based categorization with confidence scores
4. **Ticket Generation**: Structured output with proper formatting

5. **Quality Control**: Validation and review mechanisms

User Interface:

- **Dashboard**: Overview of processed feedback and generated tickets
- **Configuration Panel**: Adjust classification thresholds and priorities
- **Manual Override**: Edit or approve generated tickets
- **Analytics**: Show processing statistics and performance metrics

Output Files:

- **generated_tickets.csv**: Final ticket output with proper structure
- **processing_log.csv**: Detailed processing history and decisions
- **metrics.csv**: Performance and accuracy metrics

Step 3: Demonstration and Testing

Create a complete demo showing:

1. **Data ingestion** from your mock CSV files
2. **Real-time processing** with agent interactions
3. **Classification accuracy** compared to expected results
4. **Ticket generation** with proper formatting
5. **User interface** functionality and monitoring
6. **Error handling** and edge case management