# SMART BUDGETING AGENT

A full-stack AI system that automates financial transaction categorization

**Derrick K, Gibbs-McGlaston**
Developer and designer of the complete system

**Capstone ITAI-2277**
Final project for the AI Programmning Capstone course

Instructor: **Sitaram Ayyagari**

# Problem Statement

## Why budgeting is hard

People struggle because expense tracking is tedious and inconsistent

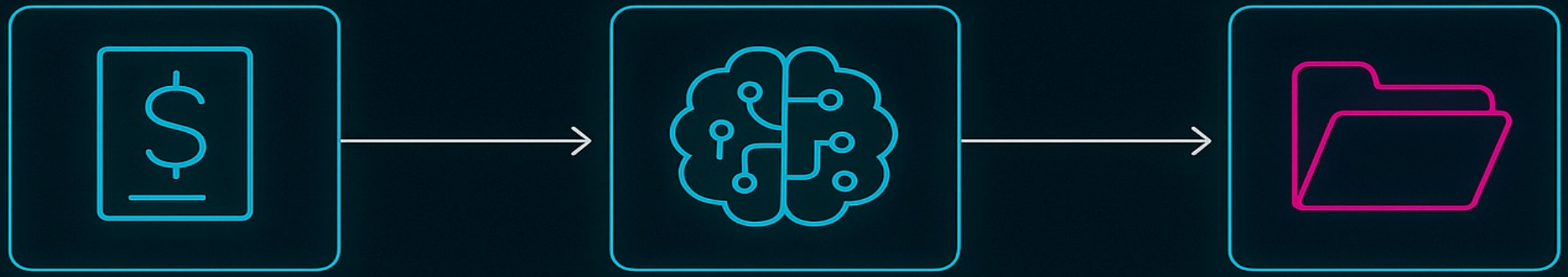
## Why categorization matters

Clean categories help users understand spending patterns and improve financial decisions

## What gap your system solves

It removes manual work by predicting categories instantly using machine learning

# EXPENSE CATEGORIZATION



Transaction

ML Model

TF-IDF vectoriz

Category prediction

Category

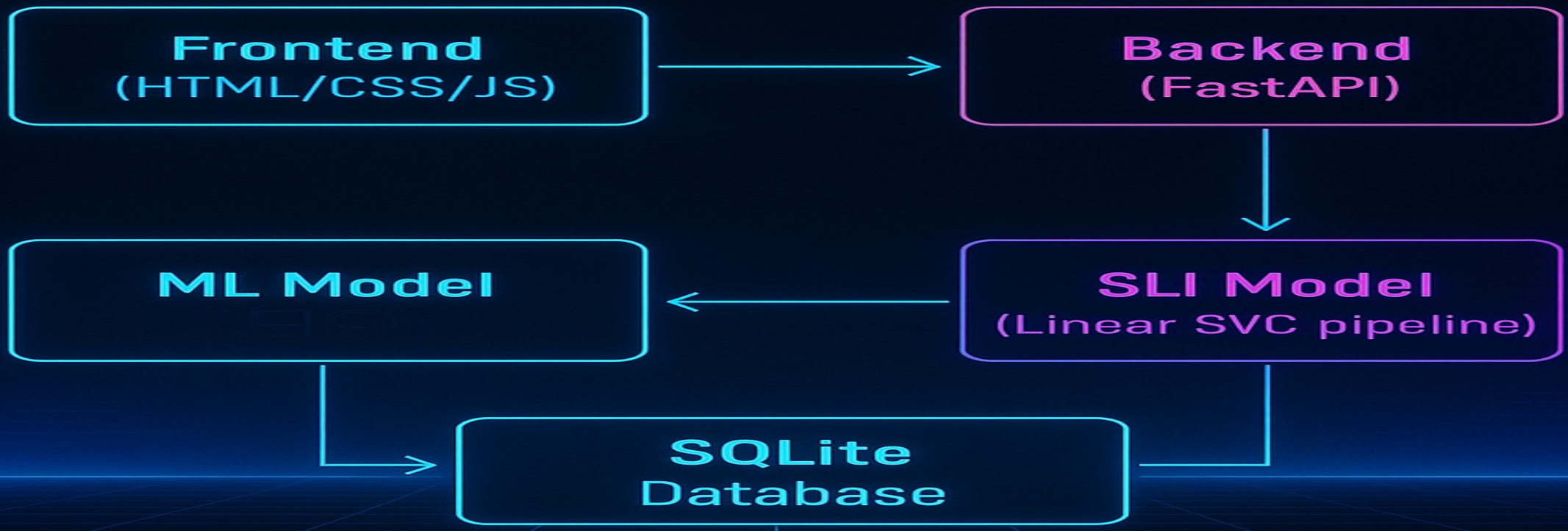
# Project Goals

**Automate expense categorization**
Replace manual labeling with a fast, accurate ML model

**Use ML + full-stack design**
Integrate machine learning with a real API and a functional UI

**Build a functional deployable system**
Deliver a prototype that can be installed, run, and tested end-to-end

# System Architecture



# DATA PIPELINE

### SOURCE OF DATA

Transactions collected and labeled for training

### CLEANING

Removed noise, punctuation, special characters, and normalized text

### TOKENIZATION

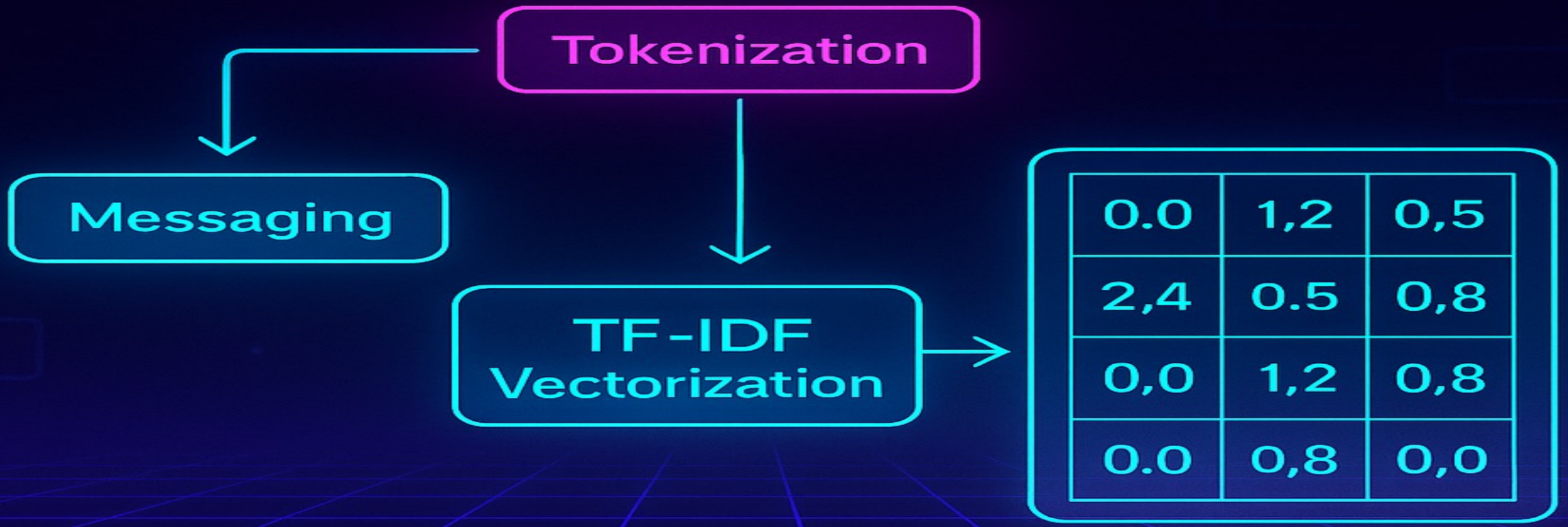
Broke text into meaningful units for ML processing

### VECTORIZATION

Converted words into TF-IDF numerical features

### TRAIN/TEST SPLIT

Ensured unbiased evaluation of model performance


Tokenization
Messaging
TF-IDF
Vectorization
0.0
1,2
0,5
2,4
0.5
0,8
0,0
1,2
0,8
0.0
0,8
0,0

# MACHINE LEARNING MODEL

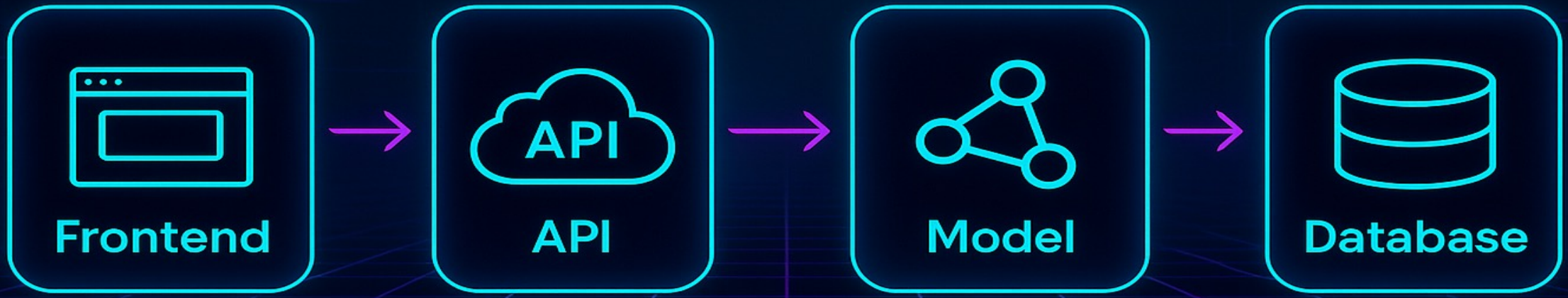
- LinearSVC — Chosen for speed and strong performance on text classification tasks
- TF-IDF — Transforms text into weighted numerical vectors the model can learn from
- Accuracy — Achieved approx. 90% accuracy on test data
- Why SVC fits the problem — It handles sparse text data extremely well with low computational cost


PREDICT
Prediction
TRANSACTION
Travel

# BACKEND API DESIGN

- /predict — Accepts transaction data and returns a predicted category
- TF-IDF — Transforms text into weighted numerical structure for clean storage
- Schema.sql — Defines the SQLIte Iatabase structure for clean storage
- Security + performance conslaiderations — Lightweight server, isolated

# REST Pipeline



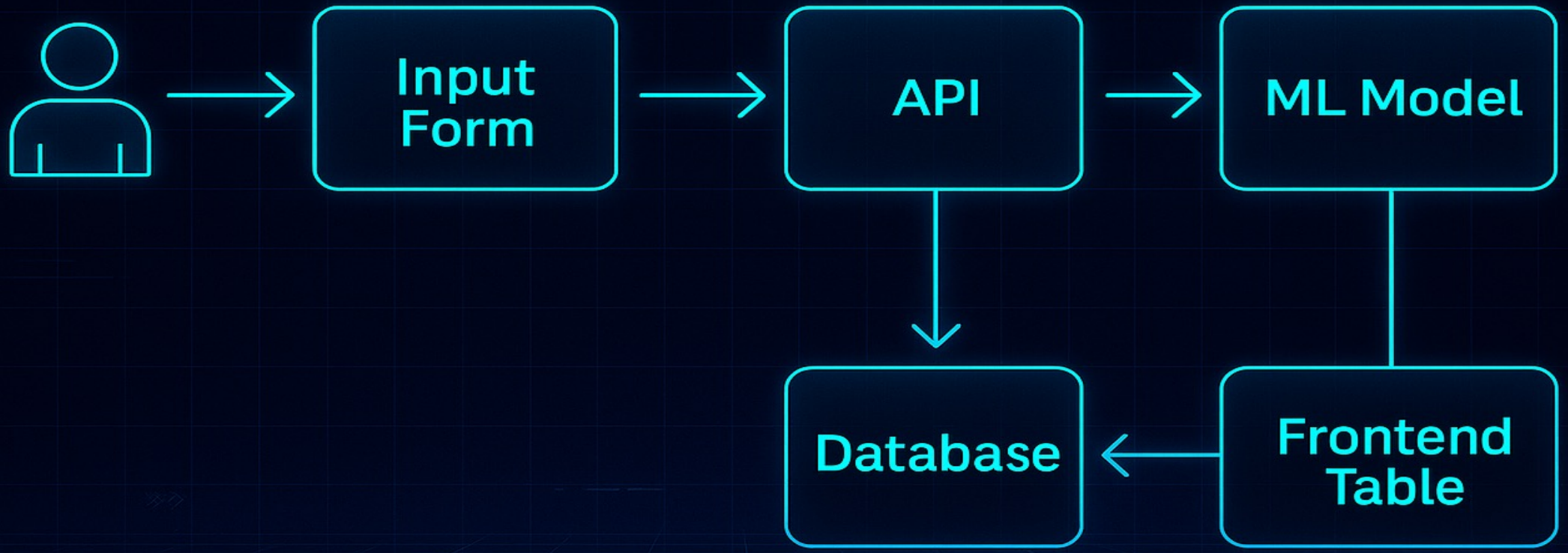
# FRONTEND INTEGRATION

- Prediction form — Users enter transaction details quickly and easily
- Live updates — Predictions show instantly without reloading the page
- Transaction table — Displays stored transactions pulled from backend
- UX simplicity — Clean, minimal design focused on functionality over visuals

# DEMO FLOW

- Enter transaction – User fills indescription, merchant, amount, etc.
- Saveive prediction – Backend returns the category predicted by the ML model
- Save to DB – The transaction and prediction are stored in SQLite
- View in recent table – Updated record appears instantly in the frontend

# PREDICTION FLOW




working.zip
smart_budgeting_mvp
Artificial Intel Resource Project
Sonic_Immortality_Spell_b...bs.docx
Capstone_Project
trinity_engine.zip
Gibbs-McGlaston Holdings
TV Shows
Other
Updates
Plex
Visa Documents
Screenshot 2025-12...05.17 PM
working
Smart_Budgeting_Agent_F...oject.zip

# TECHNICAL CHALLENGES

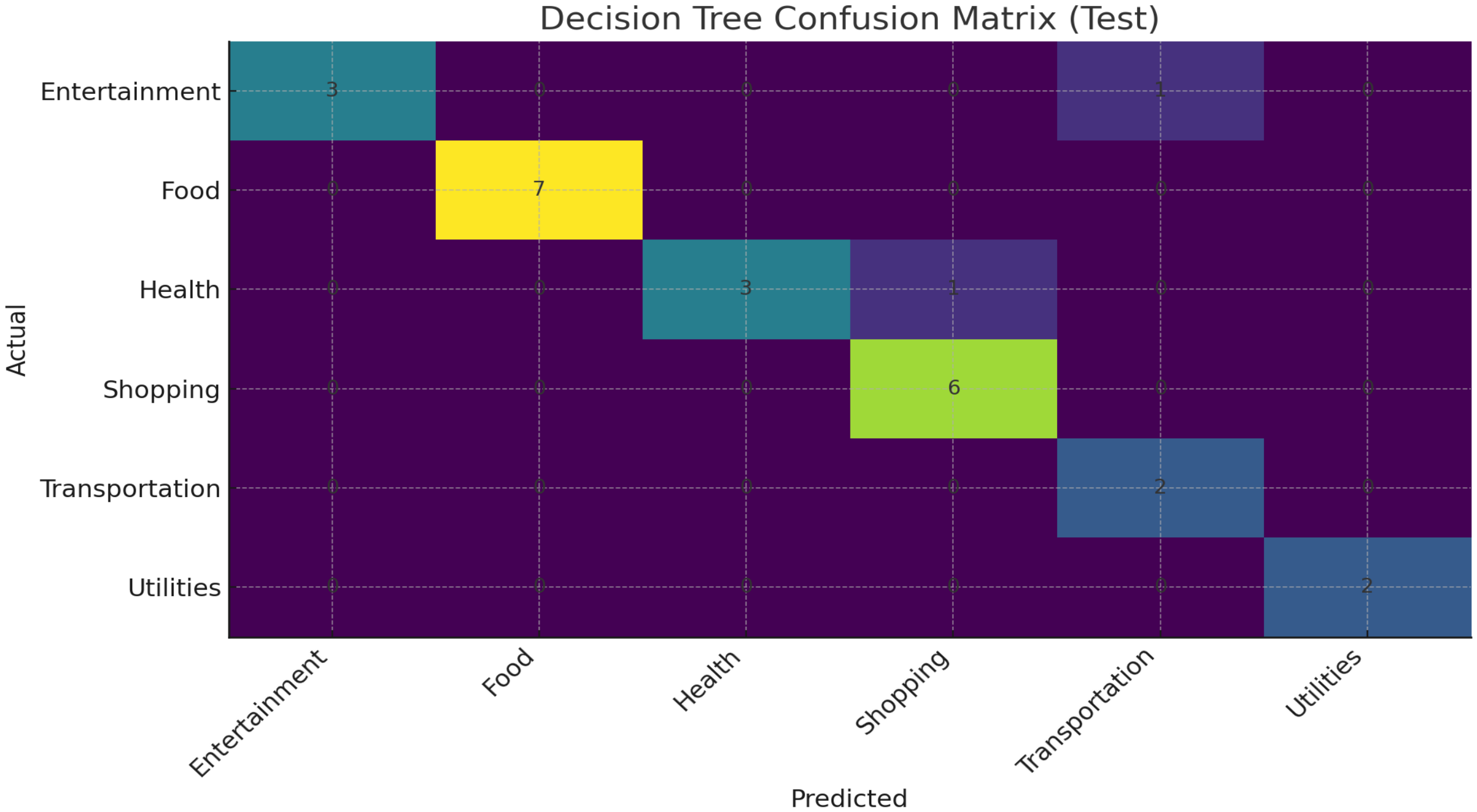
- Environment conflicts – Issues with virtual environments and dependency mistsatches
- Broken model predictions – Initial models produced inconsistent results
- Fixing label quality – Improved training data for better accuracy
- FastAPI debugging – Resolved server errors and routing issues

# SOLUTIONS IMPLEMENTED

- Rebuilt model pipeline – Created a stable reapeatable training workflow
- Added override logic – Handled cases where certain merchants map to fixed categories
- Cleaned virtual env – Removed conflicts and reinstalled dependencies cleanly
- Improved training – Refined labels and feature engineering

# EVALUATION METRICS

- Confusion matrix – Visual breakdown of model performance by class
- Added override logic – Precision, recall, and F1 scores for each category
- Accuracy – Demonstrated strong overall predictive performance
- Error patterns – Identified which categories were most difficult to classify


Decision Tree Confusion Matrix (Test)
Actual
Predicted
Entertainment
Food
Health
Shopping
Transportation
Utilities
3
1
7
3
1
6
2
2

# CAPSTONE REQUIREMENTS

- ML model – Complete ML pipeline with documentation
- Full stack integration – Frecision, recall, and Fl scores for each category
- Repo + documentation – Clean GitHub repository with full instructions
- Demo-ready prototype – Fully operational and testable

# REAL-WORLD USE CASES

- Personal finance – Individuals wanting clearer budgeting insights
- Fin-tech budgeting apps – Foundations for consumer-level financial tools
- Banking transactions – Automated categorization systems used in mobile banking
- Foundations for FussBudget – Can evolve into a full budgeting platform

# FUTURE ENHANCEMENTS

- User accounts – Add authentication and personalized dashboards
- Cloud deployment – Host backend and frontend online for public use
- Multiple models – Compare SVC with neural models for improved accuracy
- Budget alerts – Notify users when they overspend

# ETHICAL CONSIDERATIONS

- Data privacy – Ensure user data is protected and never misused
- Model bias – Monitor for biased predictions based on merchant or text patterns
- Transparency – Explain model decisions where possible
- Security considerations – Proctect API endpoints and database integrity

# LESSONS LEARNED

- ML pipeline structuring – Understanding how to properly bulld  evaluate  and save models
- Debugging real systems – Handling errors across multiple layers of a stack
- End-to-end development – Connecting data, model, backend, and frontend
- GitHub professionalisms – Publishing clean, testable, employer-ready repos

# CONCLUSION

- Restate what you built – A complete Al-powered expense categorization system
- Impact of the system – Makes budgeting easier, faster, and more accurate
- Skills demonstrated – Machine learning full-stack development, and technical problem-solving

# THANK YOU