1. Introduction & Project Context (≈3 mins)

"Hello everyone, my name is Harish Chandra, and today I'll be presenting my capstone project titled *DeepCSAT: E-Commerce Customer Satisfaction Score Prediction*.

Customer Satisfaction, or CSAT, is one of the most important metrics in the e-commerce sector. A single negative experience can make a customer abandon the platform, while a positive experience increases loyalty and the chances of repeat purchases.

Traditionally, companies have measured CSAT through post-purchase surveys. But these surveys are limited — not every customer responds, responses are delayed, and they can't always capture the full customer journey.

So, the challenge we set out to solve was: **Can we predict a customer's satisfaction score automatically, using the rich data generated during their interactions with the platform?**

The benefits are huge:

- The business can act in real time to rescue at-risk customers.
- Managers can identify patterns across agents, shifts, or product categories that affect satisfaction.
- It creates a scalable, proactive alternative to traditional surveys.

In this project, I built a complete solution:

1. A data preprocessing and feature engineering pipeline.
2. An Artificial Neural Network to predict CSAT scores.
3. A deployment-ready app built in Streamlit.

By the end of this video, you'll see how this project can directly translate into real-world improvements in customer service."

---

2. Dataset Overview (≈3 mins)

"The dataset was provided as part of the project and represents one month of customer interaction data on a platform called Shopzilla.

Let's break down the key features:

- **Service-related fields**: channel of communication (inbound, outbound, email), category, and sub-category of interaction.
- **Customer remarks**: open text feedback provided by customers. This is an unstructured field that often contains sentiment signals.
- **Order and product details**: order IDs, order timestamps, survey dates, product categories, and item prices.
- **Operational data**: connected handling time, timestamps of when an issue was reported and when it was responded to.
- **Agent information**: name, supervisor, manager, tenure bucket, and shift timing.
- **Target variable**: the CSAT score, an integer representing customer satisfaction.

Challenges included:

1. **Mixed datatypes** — numerical, categorical, text, and datetime.
2. **Missing values** — e.g., product categories not always recorded.

3. **Imbalanced labels** — extreme CSAT scores were less common.
4. **Short, noisy remarks** — text data often consisted of just a few words.

This combination makes it an excellent real-world dataset: messy, diverse, and challenging — exactly the kind of problem deep learning can shine on."

---

3. Data Cleaning & Feature Engineering (≈4 mins)

"Data preparation is where most of the effort in machine learning projects lies. For this dataset, I applied several careful preprocessing steps:

1. **Missing values**: instead of discarding rows, I flagged missing data. For example, if item price was missing, I created a binary feature called *missing_Item_Price*. This helps the model learn if the absence of information is itself meaningful.
2. **Feature engineering**:
   - *Response delay* — the time in hours between issue reported and issue responded. This captures responsiveness, which we expect to correlate with satisfaction.
   - *Item price binning* — grouping prices into ranges like 0–100, 100–500, etc. This captures broad spending segments while reducing sensitivity to outliers.
   - *Temporal features* — from timestamps, I extracted day of week and hour of day. For example, issues reported on weekends may have different satisfaction patterns.
3. **Categorical encoding**:
   - Low-cardinality features (like agent shift, tenure bucket) were one-hot encoded.
   - High-cardinality features like agent name were excluded from the first version, since they risked overfitting. Instead, aggregate features like *average CSAT per agent* could be explored in future iterations.
4. **Customer remarks preprocessing**:
   - Cleaned text: lowercased, removed punctuation, removed stopwords.
   - Converted remarks into numerical features using TF-IDF or embeddings. For example, words like *"delay"*, *"refund"*, *"bad"* often point to dissatisfaction, while *"thank you"* or *"great service"* point to higher scores.

All transformations were wrapped into a **feature engineering pipeline** and saved with joblib. This is important because at prediction time, we need to apply the exact same transformations to new data as we did during training.

By the end of preprocessing, each interaction record was converted into a vector of features that combined structured metadata and semantic signals from text."

---

4. Model Development (≈4 mins)

"With features prepared, I moved on to designing the model.

The task was framed as a **multi-class classification problem**, predicting the CSAT score class.

The chosen architecture was an **Artificial Neural Network (ANN)** because:

- It can model nonlinear interactions.
- It easily integrates text embeddings with structured features.

- It is computationally efficient for deployment.

The architecture:

- Input layer for structured features + text embeddings.
- Two to four hidden layers with ReLU activation.
- Dropout and L2 regularization to reduce overfitting.
- Output layer with softmax activation to produce probabilities for each CSAT class.

Training strategy:

- Optimizer: Adam with a carefully tuned learning rate.
- Early stopping on validation loss to prevent overfitting.
- Class imbalance handled using class weights, so rare CSAT scores still influenced learning.
- Stratified splits ensured training, validation, and test sets had similar class distributions.

I also explored alternative formulations:

- Regression, treating CSAT as continuous.
- Ordinal regression, to model ordering explicitly.
  Ultimately, classification performed best and aligned with how businesses interpret discrete CSAT scores."

---

5. Evaluation & Insights (≈3 mins)

"Evaluation was multi-pronged.

- **Accuracy**: to measure overall correctness.
- **Macro F1-score**: to balance performance across all classes.
- **Mean Absolute Error (MAE)**: to measure average distance between predicted and true scores, treating CSAT as ordinal.

The model achieved strong performance, with high accuracy and macro-F1. Most errors were minor — predicting a 3 when the true score was 4 — rather than extreme misclassifications.

The confusion matrix confirmed this, showing that errors clustered around adjacent scores.

Using SHAP for interpretability, I identified top drivers of CSAT:

- **Response delay** — longer delays strongly predicted lower CSAT.
- **Customer remarks sentiment** — negative language pushed predictions downward.
- **Agent tenure and shift** — experienced agents, especially in morning shifts, tended to correlate with higher satisfaction.

These insights provide **direct, actionable recommendations**: invest in faster response handling and focus training on shifts or agents where dissatisfaction is concentrated."

---

6. Walkthrough of the Streamlit Interface (NEW – ≈3 mins)

"To make the model accessible to non-technical users, I built a Streamlit app. Let me walk you through it.

- At the top, we select interaction details such as channel, category, and sub-category.
- In the middle, we can enter **customer remarks** — for example, *'My refund is delayed and no one is responding.'*
- Below, we provide order metadata such as order ID, order date, survey date, and timestamps for issue reported and responded.
- We also enter product information — category, price, handling time.
- Finally, we provide agent-related data — tenure bucket and shift.

Once all fields are entered, we click *Predict CSAT*.

What happens behind the scenes:

1. The raw input is converted into a DataFrame.
2. The same preprocessing pipeline from training is applied — encoding categories, computing response delay, binning prices, vectorizing text.
3. The ANN model is called to make a prediction.
4. The output is decoded using the label encoder and displayed as a CSAT score.

For example, for a delayed refund with negative remarks, the model might predict a CSAT of 2. For a smooth purchase with positive feedback, it might predict a 5.

The app provides instant results, making it usable by customer service managers who want quick insights without touching Python code."

---

7. Deployment & Real-World Integration (≈2 mins)

"For deployment, I followed best practices:

- Saved artifacts: preprocessing pipeline, ANN model, label encoder.
- Reused the exact same pipeline during inference for consistency.
- Used Streamlit for a quick, user-friendly interface.

In a real-world setting, this app could be integrated into a CRM system. For example, whenever a new interaction record is logged, the system could call the model and predict a CSAT score immediately.

Managers could get a **dashboard of at-risk customers** and take proactive measures such as priority callbacks, refunds, or special offers.

For scalability, this could be containerized using Docker and deployed on cloud services like AWS or GCP. If low-latency is required, TensorFlow Serving or FastAPI could replace Streamlit."

---

8. Limitations, Ethics & Future Work (≈2 mins)

"Of course, the project has some limitations:

- Dataset is only one month, which may miss seasonal trends.
- Some categories are underrepresented.
- Text data is short and noisy.

From an ethical perspective, predictions should be used responsibly — to improve customer experience, not to penalize individual employees unfairly. Personally identifiable information must be anonymized.

Future work includes:

- Using **ordinal regression** to model the ordered nature of CSAT more explicitly.
- Leveraging **transformer-based embeddings** for customer remarks, which could capture context more effectively.
- Incorporating historical agent-level or customer-level trends.
- Running **A/B tests** to measure the actual business impact of using predictions in real time."

---

9. Conclusion & Call-to-Action (≈2 mins)

"In summary, *DeepCSAT* demonstrates how deep learning can enhance e-commerce operations by predicting customer satisfaction automatically.

We built a full pipeline: from cleaning messy data, engineering meaningful features, training a robust neural network, and deploying it in a user-friendly app.

The model not only predicts CSAT with strong accuracy but also uncovers actionable insights — like the importance of fast responses and experienced agents.

This project shows the real-world potential of AI: moving beyond academic exercises to solving business-critical problems.

All code, notebooks, and the app are documented in the GitHub repository [Insert GitHub Link].

Thank you for watching this presentation. I look forward to any questions or feedback you might have."