Smartwatch Sentiment Analyzer

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology

In

Computer Science and Engineering School of Engineering and Sciences

Submitted by

B.Murali Krishna	AP23110010656
M.Thavanesh Krishna	AP23110010619
K.Mahasvin	AP23110010075
K.Harshavardhini	AP23110011144
B.Sai Charan	AP23110010173

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[December,2025]

Smartwatch Sentiment Analyzer

Project Description:

The Smartwatch Sentiment Analyzer is an end-to-end machine learning and NLP project designed to automatically detect customer sentiment from smartwatch product reviews.

The system processes user reviews taken from e-commerce platforms (such as Amazon gadget reviews), cleans the text, and classifies each review as Positive, Negative, or Neutral.

Three different approaches are used:

- A baseline rule-based model using TextBlob and domain-specific keyword rules
- A classical machine-learning model based on TF–IDF features and Logistic Regression
- A Transformer-based model using a fine-tuned DistilBERT language model

The project also includes a Flask web application where users can:

- Log in or sign up
- Enter a single smartwatch review and see its sentiment and aspect-wise analysis
- Upload a CSV file of many reviews and get a batch sentiment summary
- Access REST APIs for integration with other applications

By combining traditional ML, modern transformers, and a practical web interface, the Smartwatch Sentiment Analyzer demonstrates how AI can help companies understand product feedback and improve smartwatch design, features, and customer experience.

Project Scenarios

Scenario 1: Individual Customer Review Analysis

A smartwatch owner wants to know whether their written review sounds positive or negative before posting it online. They log into the web application, paste their review into the text box, and click "Analyze Review".

Within seconds, the system:

- Checks whether the text really describes a smartwatch
- Predicts the overall sentiment (Positive/Negative/Neutral)
- Highlights sentiment strength (confidence score)
- Provides aspect-wise insights, e.g., Battery, Display, Fitness Tracking, Comfort, Price

This helps the user understand how their review may be interpreted and which aspects are praised or criticized.

Scenario 2: Brand Monitoring for a Specific Smartwatch

A smartwatch manufacturer wants to monitor opinions about its latest model. The marketing team exports recent Amazon reviews into a CSV file and uploads it through the Batch Analysis page.

The system processes hundreds of reviews automatically:

- Filters out reviews that are not really about smartwatches
- Classifies each relevant review into Positive/Neutral/Negative
- Calculates overall sentiment distribution
- Computes the average polarity score and shows example comments

The team can quickly see whether recent updates improved user satisfaction and which issues (like battery life or strap quality) are mentioned frequently.

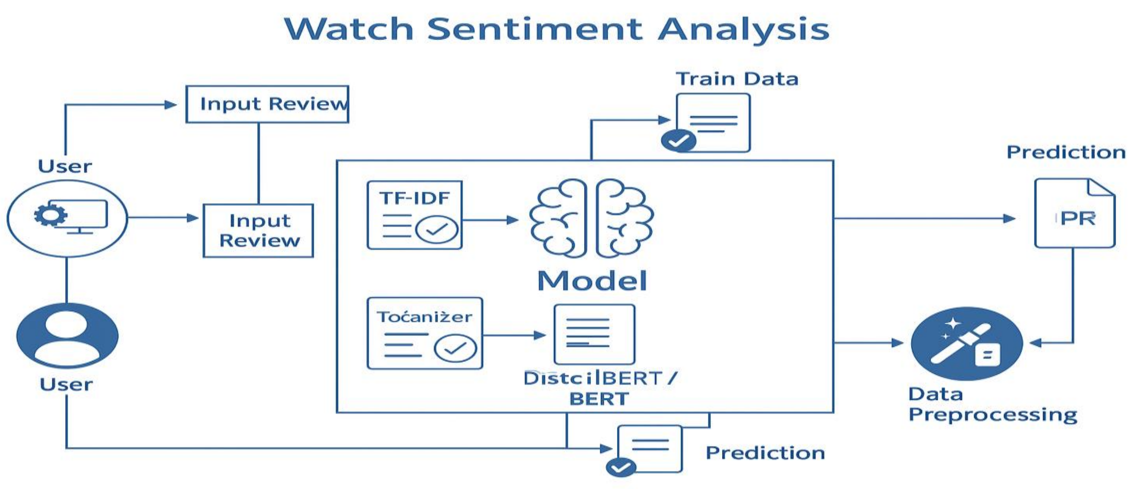
Scenario 3: Comparative Study of ML vs Transformer Models

A data science team wants to compare older ML methods with contextual transformer models. Using the same labeled smartwatch review dataset, they run the classical training script and the transformer training script.

The comparison module evaluates both models on the same test split and prints accuracy, classification reports, and confusion matrices.

The team observes how DistilBERT handles slang, context, and subtle phrases better than the TF–IDF Logistic Regression model, proving the advantage of transformer-based architectures for real-world sentiment analysis.

Technical Diagram:



Prerequisites

To complete and run this project, the following tools and packages are required:

Software

- Anaconda / Python 3.x environment
- Code editor / IDE: VS Code, PyCharm, or Jupyter Notebook
- Web browser (for using the Flask web interface)

Python Packages Required:

Install these packages in your environment (for example using pip install -r requirements.txt):

- numpy – numerical operations
- pandas – dataset handling and CSV I/O
- scikit-learn – classical ML algorithms and evaluation metrics
- torch – PyTorch backend for transformer models
- transformers – Hugging Face models (DistilBERT)
- datasets – dataset handling with Hugging Face
- textblob – baseline sentiment analysis and sentence tokenization
- flask – web framework
- joblib – saving/loading classical models and transformers
- matplotlib / seaborn – optional for graphs
- sqlite3 (built into Python) – authentication database

Prior Knowledge

Before working on this project, you should have basic understanding of:

Machine Learning Concepts

● Supervised Learning (classification)

https://www.javatpoint.com/supervised-machine-learning

● Text Preprocessing & NLP basics:

Tokenization, stopwords, TF-IDF.

● TF-IDF Vectorization

https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

● Logistic Regression (Classical ML Model)

https://www.javatpoint.com/logistic-regression-in-machine-learning

● Evaluation Metrics:

Accuracy, Precision, Recall, F1-score

https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/

● Transformer Models (BERT)

https://huggingface.co/docs/transformers/index

● Flask Basics:

https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Flow:

1. User or developer collects smartwatch review data from Amazon gadget reviews or similar sources.
2. Data is cleaned and labelled (Positive/negative/neutral) stored as

   smartwatch_labelled.csv
3. Exploratory Data Analysis (EDA) is performed to understand label distribution, review lengths, and word usage.
4. Baseline, classical, and transformer models are built:
   - TextBlob + domain rules (baseline)
   - TF–IDF + Logistic Regression (classical)
   - DistilBERT fine-tuned for sentiment (transformer)
5. Performance testing is done on a common test set, and metrics are compared.
6. Best models are saved and integrated into the system.
7. Flask web application is used by end users to:
   - Analyze a single review
   - Perform batch analysis on uploaded CSVs
   - Consume REST APIs for integration
8. Results and insights are used by product teams and researchers to understand customer satisfaction and improve smartwatch features.

Project Activities

Activity 1: System Design

- Designed UI flow for login, single analysis, and batch analysis
- Defined core functions: relevance check, sentiment scoring, aspect extraction

- Designed folder structure and templates

Activity 2: Building the Sentiment Engine

- Implemented TextBlob-based polarity detection
- Defined thresholds for Positive, Negative, and Neutral
- Created domain rules for smartwatch-specific corrections
    - e.g., “battery draining fast” → Negative
    - “battery lasts all day” → Positive

Activity 3: Smartwatch Relevance Detection

- Created keyword dictionary
- Implemented a function to reject irrelevant text unless user overrides
- Useful to avoid analyzing unrelated or random sentences

Activity 4: Aspect-Based Analysis

- Built keyword groups for each aspect
- Extracted aspect-related sentences from the review
- Computed polarity and sentiment separately for each aspect

Activity 5: Web Application Development

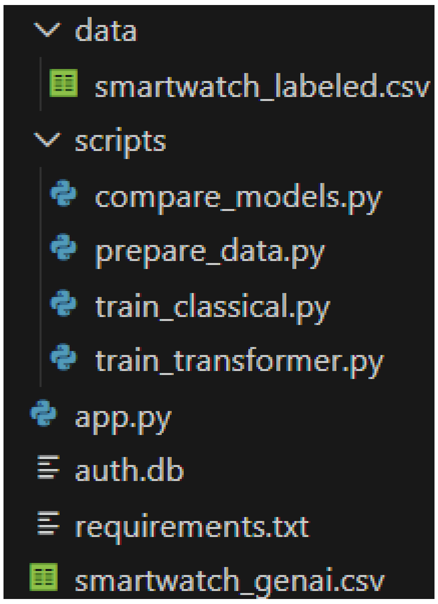
- Login & signup using SQLite
- Single review analyser page
- Batch review analyser page
- REST API endpoints
- Bootstrap/HTML-based user interface

Activity 6: Batch Processing

- Allowed CSV uploads
- Automated text column detection
- Analyzed each row
- Generated summary statistics and displayed clean tables

Project Structure:

Create the Project folder which contains files as shown below



```
data
    smartwatch_labeled.csv
scripts
    compare_models.py
    prepare_data.py
    train_classical.py
    train_transformer.py
app.py
auth.db
requirements.txt
smartwatch_genai.csv
```

Project Structure Explanation:

- app.py
  Flask web application (routes, authentication, front-end integration, API endpoints).

- prepare_data.py
  script indicating how raw smartwatch reviews are cleaned and converted into a labeled dataset.

- train_classical.py
  trains the TF–IDF + Logistic Regression model and saves classical_model.pkl and tfidf_vectorizer.pkl.

- train_transformer.py
  fine-tunes DistilBERT, evaluates it, and saves the model under models/transformer distilbert

- compare_models.py
  loads the labeled dataset and both models to compare their performance on the same test split.

- data/
  - smartwatch_labeled.csv – cleaned and labeled smartwatch reviews.

- templates/
  - single.html or inline templates – page for single review analysis.
  - batch.html – page for CSV batch analysis.

    - login.html, signup.html – authentication pages.

- static/
    - CSS styles for the modern, neon-style UI.
    - Optional JavaScript files for front-end enhancements.

- auth.db – SQLite database for user accounts.
- requirements.txt – list of Python dependencies.

Milestone 1: System Design & Requirements Analysis

Activity 1.1: Understanding the Problem

Smartwatch reviews contain opinions about features such as battery life, display, fitness tracking, comfort, notifications and price.The goal is to automatically extract meaningful sentiment (Positive/Negative/Neutral) from such user-written reviews without using any dataset or machine-learning training.

Therefore, the system must:

- Accept user input (single review or CSV file)
- Detect if the input text is smartwatch-related
- Compute sentiment using TextBlob
- Apply custom domain rules to improve accuracy for smartwatch terminology
- Analyze aspect-wise sentiment
- Display results in a clean web interface

Activity 1.2: System Architecture Planning

The system is designed around three main components:

1. **Sentiment Engine**
    - TextBlob polarity extraction
    - Domain-specific rule adjustments
    - Smartwatch relevance detection
    - Aspect-based sentiment extraction
2. **Flask Web Application**
    - Login / signup with SQLite
    - Single review analysis
    - Batch CSV review analysis
    - REST APIs for external systems
3. **User Interface (Frontend)**
    - HTML templates
    - Clean, card-based layout
    - Result cards and polarity meters

Activity 1.3: Functional Requirements

- User authentication (login/signup)
- Enter or paste a review for analysis
- Smartwatch relevance checking
- Sentiment prediction with polarity & confidence
- Aspect-level insights (battery, display, comfort, design, price, notifications, tracking)
- Upload CSV for batch analysis
- API endpoints for integration

Activity 1.4: Non-Functional Requirements

- Lightweight (no large models or datasets)
- Fast response time
- Clean and minimal UI
- Easily deployable on any machine
- Accurate for smartwatch-specific reviews due to rules

Activity 1.5: Technology Stack

- Python 3.x
- Flask - backend web framework
- TextBlob - rule-based sentiment engine
- Pandas - CSV processing
- SQLite - authentication database
- HTML/CSS - frontend templates

Activity 1.6: Finalized System Design

A clean feedback processing pipeline is created:

User Input → Relevance Detector → TextBlob Polarity → Domain Rule Engine → Aspect Analyzer → Display Results

For batch mode:

CSV Input → Column Identification → Row-wise Analysis → Summary Generator → Output Table

Milestone 2: Development of the Sentiment Engine

Activity 2.1: Smartwatch Relevance Detection

Before analyzing any text, the system checks whether it actually refers to smartwatch topics. This prevents users from running sentiment analysis on unrelated text accidentally.

The detector uses keyword groups such as:

- smartwatch, watch, wearable, strap
- battery, charging, drain
- screen, display, AMOLED
- heart rate, steps, fitness, sensor

If none of these appear, the system warns the user unless they choose "Proceed Anyway".

Activity 2.2: TextBlob Polarity Extraction

TextBlob is used as the core sentiment engine:

- Polarity ∈ [-1, 1]
- Polarity > 0.05 → Positive
- Polarity < -0.05 → Negative
- Otherwise → Neutral

This forms the baseline sentiment.

2.3 Domain-Specific Rule Engine

To improve accuracy for smartwatch reviews, custom rules are applied:

Negative Triggers

- "battery draining fast"
- "overheating"
- "strap broke"
- "screen cracked"
- "inaccurate tracking"

These immediately force Negative even if TextBlob finds weak polarity.

Positive Triggers

- "battery lasts all day"

- “very accurate tracking”
- “excellent display”
- “super comfortable”
- “fast charging”

These force Positive sentiment.

Activity 2.4: Confidence Scoring

A simple confidence measure uses:

confidence = |polarity| * 100

This helps users know how strong or uncertain the detected sentiment is.

Activity 2.5: Aspect-Based Sentiment Extraction

To give deeper insights, reviews are analyzed for aspects:

- Battery
- Display
- Comfort
- Fitness Tracking
- Notifications
- Design & Build
- Price

For each aspect:

1. Identify sentences containing aspect keywords
2. Run TextBlob on those sentences
3. Apply domain rules
4. Return aspect-level polarity + sentiment

Milestone 3: Web Application Development

Activity 3.1: Flask Web Application Development

Frontend (HTML Templates)

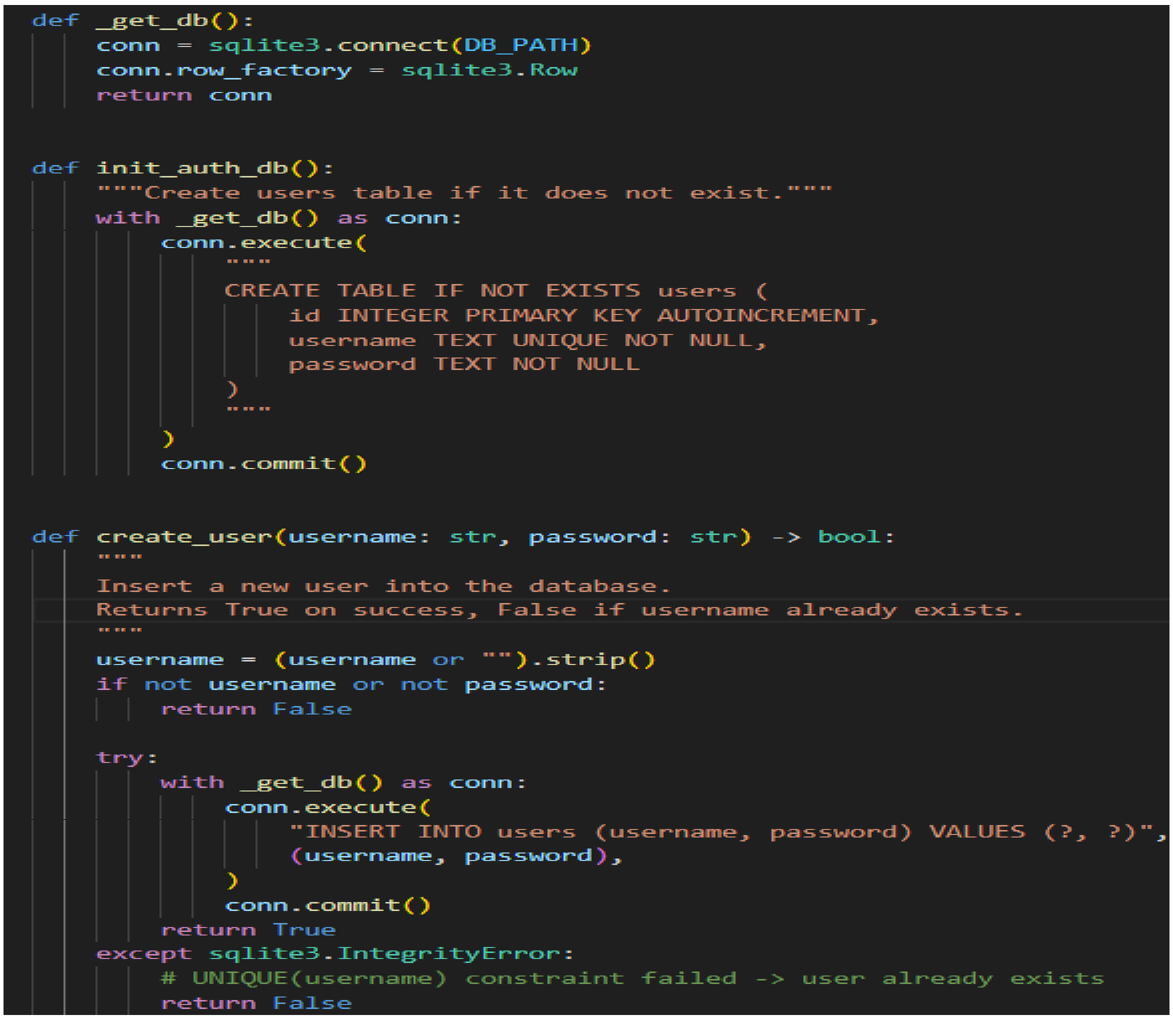
- Modern, card-based UI with gradient background.
- Single review page:
    - Text area for input, checkbox to override smartwatch relevance filter, and "Analyze review" button.
    - Output cards show smartwatch relevance, sentiment, polarity and confidence with a horizontal polarity bar.
    - Aspect list provides per-aspect labels and evidence sentences.
- Batch analysis page:
    - File upload section.
    - Optional text column name selection.
    - Summary cards (total reviews, relevant reviews, sentiment counts, average polarity).
    - Scrollable table showing review text, sentiment, polarity, confidence and relevance flag.

Backend (app.py)

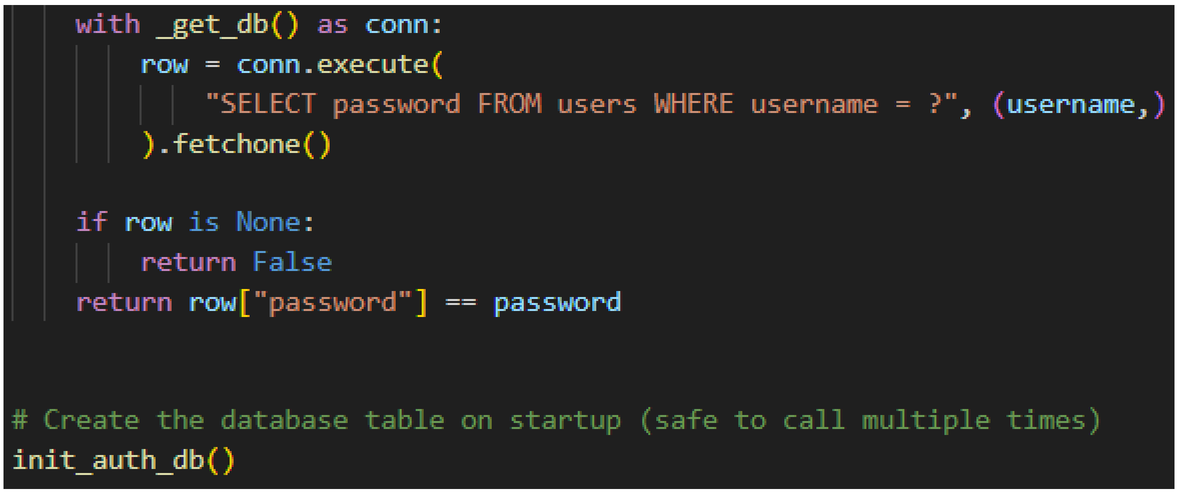
- **Authentication**
    - SQLite database auth.db for storing usernames and passwords.
    - Routes for /signup, /login, and /logout.
- **Single Review Analysis Route (/)**
    - Logged-in users can input a review.
    - The system checks whether the text is smartwatch related.
    - If allowed, TextBlob + domain rules compute sentiment, polarity and confidence.
    - Aspect analysis module scans for keywords related to battery, display, comfort, design, fitness tracking, notifications and price, and provides aspect-wise sentiment.
- **Batch Analysis Route (/batch)**
    - Users upload a CSV file of reviews.

- The server identifies the text column, filters non-smartwatch reviews, and analyzes the rest.
- Returns a detailed table of results and a summary of sentiment distribution.

- **REST APIs**
  - /api/health – health check.
  - /api/analyze-review – accepts JSON with review text and returns sentiment, confidence and aspect details.
  - /api/analyze-batch – accepts CSV file and returns summary and detailed results in JSON.



```python
def _get_db():
    conn = sqlite3.connect(DB_PATH)
    conn.row_factory = sqlite3.Row
    return conn


def init_auth_db():
    """Create users table if it does not exist."""
    with _get_db() as conn:
        conn.execute(
            """
            CREATE TABLE IF NOT EXISTS users (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                username TEXT UNIQUE NOT NULL,
                password TEXT NOT NULL
            )
            """
        )
        conn.commit()


def create_user(username: str, password: str) -> bool:
    """
    Insert a new user into the database.
    Returns True on success, False if username already exists.
    """
    username = (username or "").strip()
    if not username or not password:
        return False

    try:
        with _get_db() as conn:
            conn.execute(
                "INSERT INTO users (username, password) VALUES (?, ?)",
                (username, password),
            )
            conn.commit()
        return True
    except sqlite3.IntegrityError:
        # UNIQUE(username) constraint failed -> user already exists
        return False
```



```
    with _get_db() as conn:
        row = conn.execute(
            "SELECT password FROM users WHERE username = ?", (username,)
        ).fetchone()

    if row is None:
        return False
    return row["password"] == password


# Create the database table on startup (safe to call multiple times)
init_auth_db()
```

The Flask backend handles the core ML workflow for predicting sentiment from smartwatch reviews:

- **Model Loading:**
    - Loads the serialized classical ML model (tfidf_model.pkl) and optionally the Transformer model if deployed.
    - Model and vectorizer are loaded on application startup to minimize prediction latency.

Data Handling:

- Parses the review text submitted via a POST request from the web form.
- **Prediction:**
    - Uses `clf.predict()` for predicted sentiment (positive/negative).
    - `clf.predict_proba()` gives a confidence score (e.g., "92.45% Confidence").
- **Result Interpretation:**
    - Maps model output to user-friendly labels: "Positive Sentiment" or "Negative Sentiment."

Frontend Implementation (HTML Templates)

The web interface is designed for clarity and ease of use, optimized for general users evaluating smartwatch reviews.

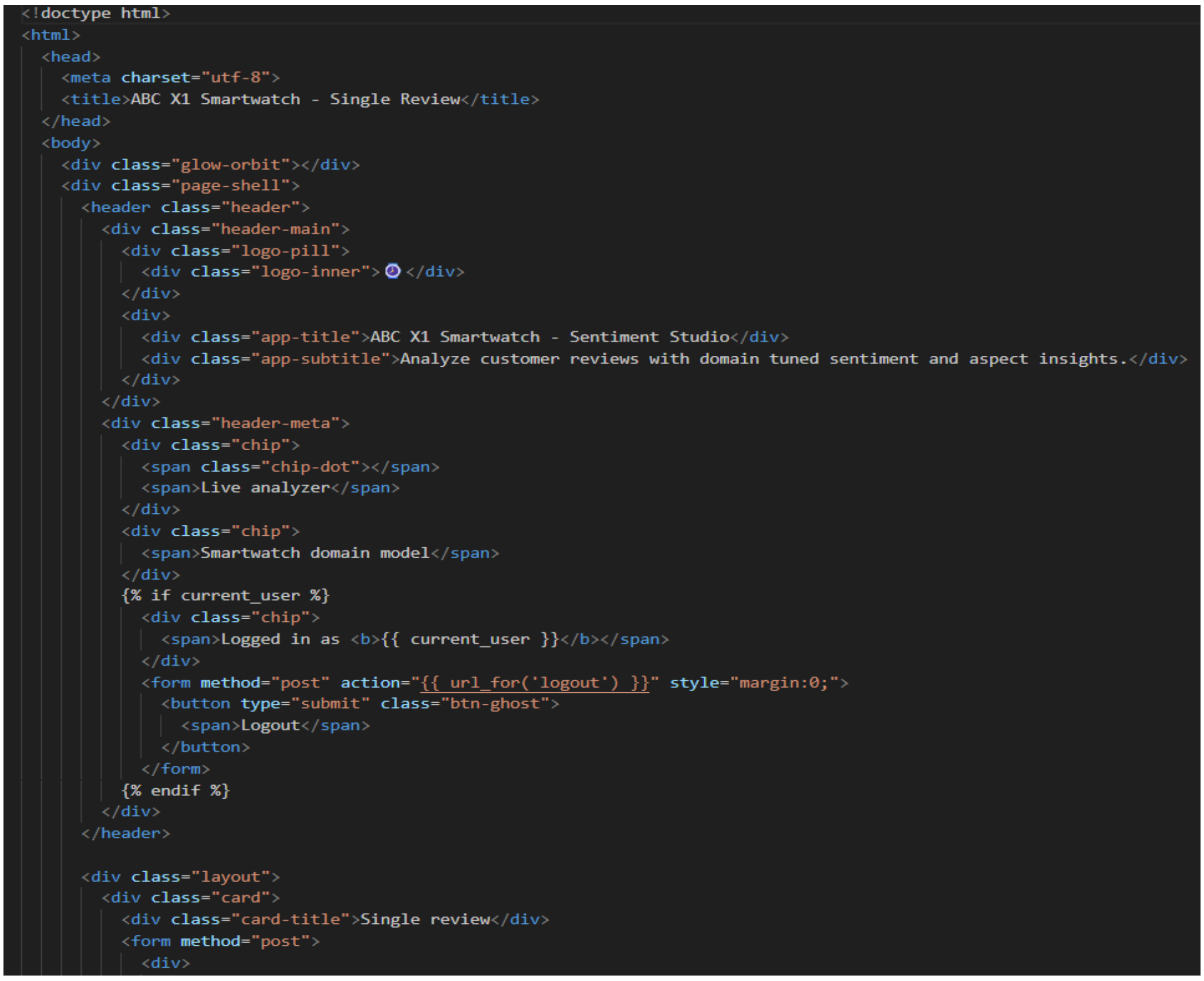
Key Features of the Web Application

1. **User Interface Components:**
    - Professional header with app branding.
    - Input form for entering smartwatch reviews.

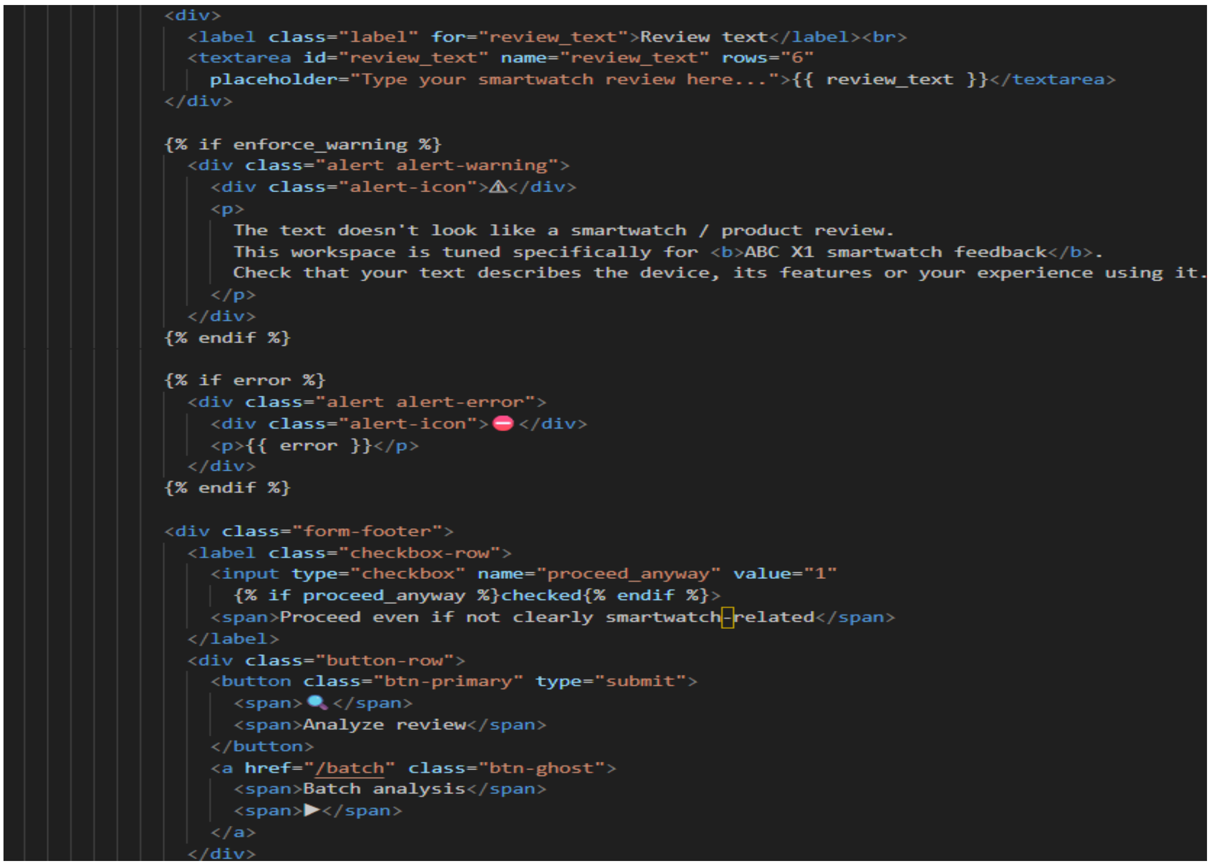
   - Mobile-responsive design using CSS for universal access.

2. **Sentiment Analysis Output:**
   - AI-powered sentiment prediction.
   - Confidence score displayed as a percentage.
   - Example: "Positive Sentiment – 93.12% Confidence"
3. **Safety & Validation Features:**
   - Input validation ensures that the review text is non-empty.
   - Secure handling of user inputs via Flask backend.

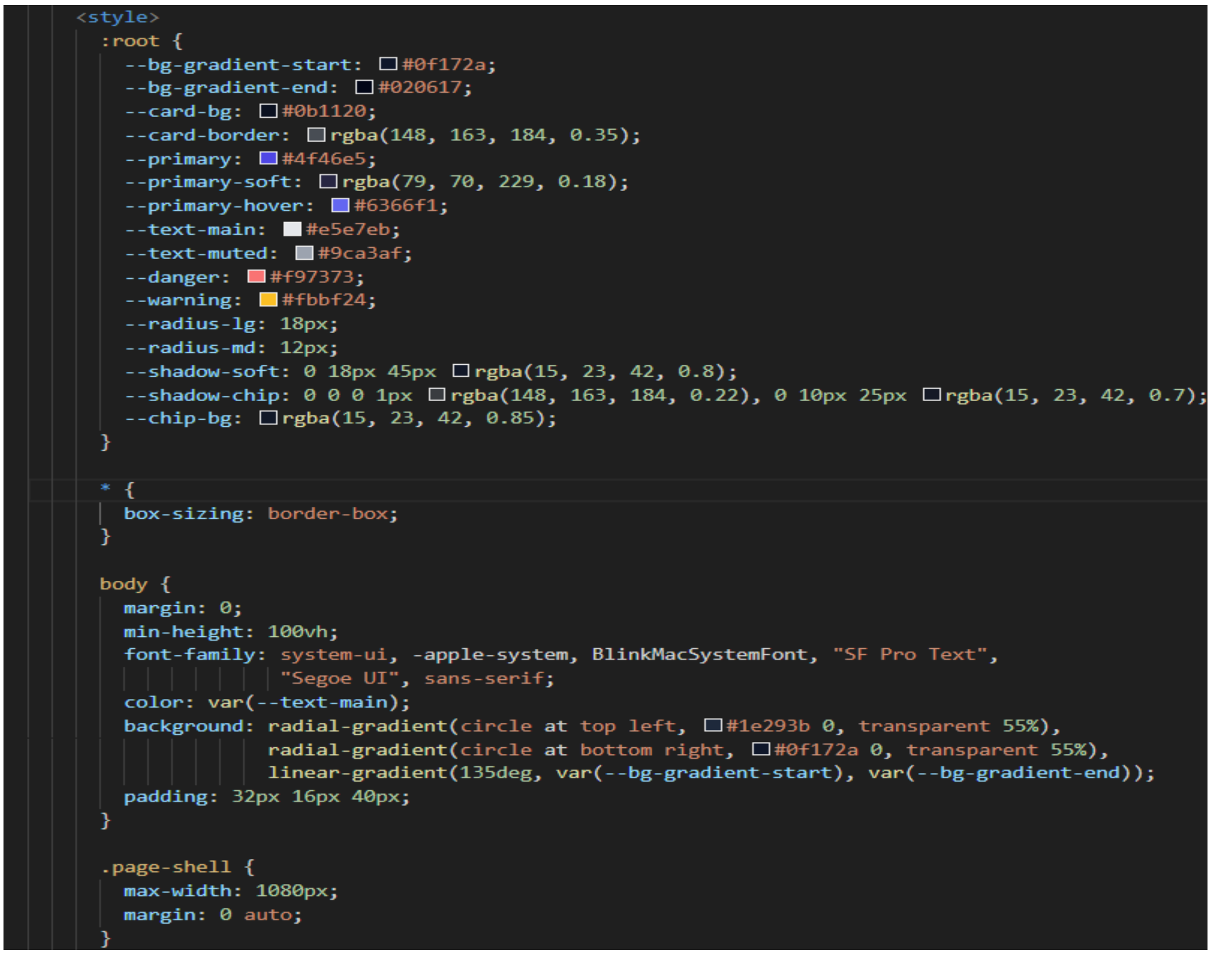
Application Screenshots



```html
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>ABC X1 Smartwatch - Single Review</title>
  </head>
  <body>
    <div class="glow-orbit"></div>
    <div class="page-shell">
      <header class="header">
        <div class="header-main">
          <div class="logo-pill">
            <div class="logo-inner"> </div>
          </div>
          <div>
            <div class="app-title">ABC X1 Smartwatch - Sentiment Studio</div>
            <div class="app-subtitle">Analyze customer reviews with domain tuned sentiment and aspect insights.</div>
          </div>
        </div>
        <div class="header-meta">
          <div class="chip">
            <span class="chip-dot"></span>
            <span>Live analyzer</span>
          </div>
          <div class="chip">
            <span>Smartwatch domain model</span>
          </div>
          {% if current_user %}
            <div class="chip">
              <span>Logged in as <b>{{ current_user }}</b></span>
            </div>
            <form method="post" action="{{ url_for('logout') }}" style="margin:0;">
              <button type="submit" class="btn-ghost">
                <span>Logout</span>
              </button>
            </form>
          {% endif %}
        </div>
      </header>

      <div class="layout">
        <div class="card">
          <div class="card-title">Single review</div>
          <form method="post">
            <div>
```



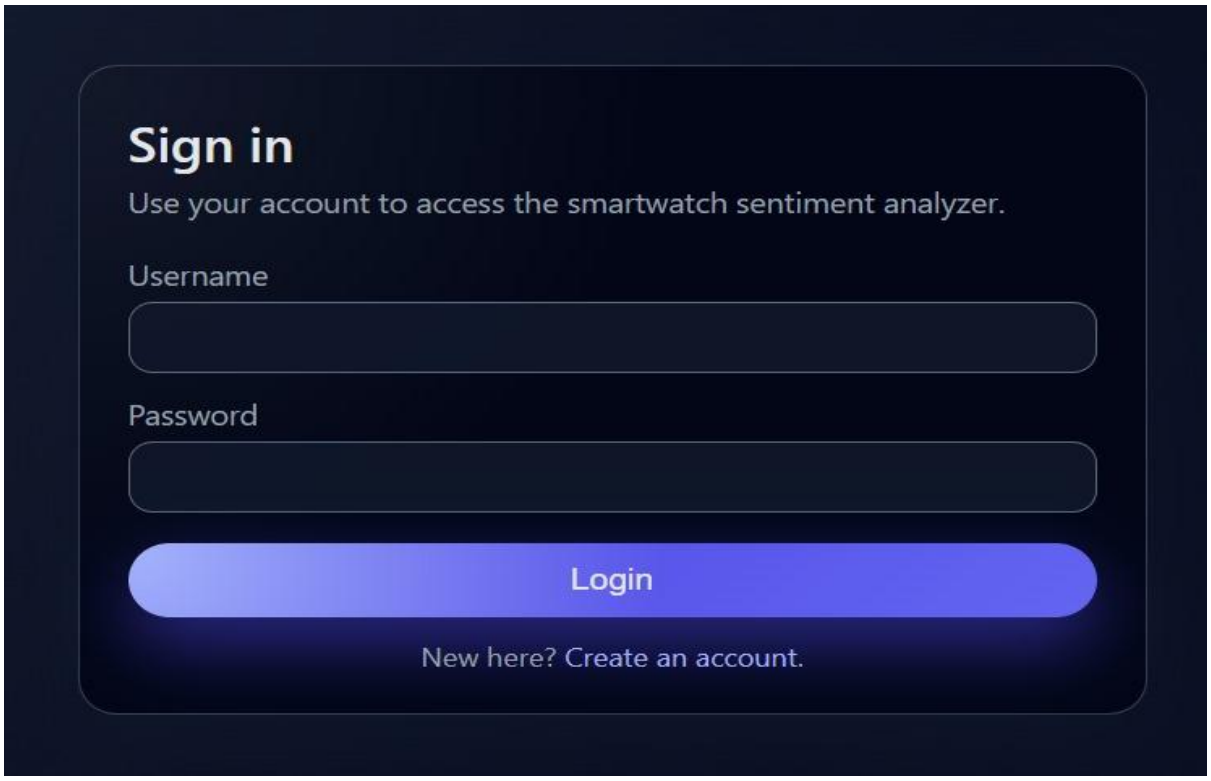
```
<div>
  <label class="label" for="review_text">Review text</label><br>
  <textarea id="review_text" name="review_text" rows="6"
    placeholder="Type your smartwatch review here...">{{ review_text }}</textarea>
</div>

{% if enforce_warning %}
  <div class="alert alert-warning">
    <div class="alert-icon">⚠</div>
    <p>
      The text doesn't look like a smartwatch / product review.
      This workspace is tuned specifically for <b>ABC X1 smartwatch feedback</b>.
      Check that your text describes the device, its features or your experience using it.
    </p>
  </div>
{% endif %}

{% if error %}
  <div class="alert alert-error">
    <div class="alert-icon">⛔</div>
    <p>{{ error }}</p>
  </div>
{% endif %}

<div class="form-footer">
  <label class="checkbox-row">
    <input type="checkbox" name="proceed_anyway" value="1"
      {% if proceed_anyway %}checked{% endif %}>
    <span>Proceed even if not clearly smartwatch-related</span>
  </label>
  <div class="button-row">
    <button class="btn-primary" type="submit">
      <span>🔍</span>
      <span>Analyze review</span>
    </button>
    <a href="/batch" class="btn-ghost">
      <span>Batch analysis</span>
      <span>▶</span>
    </a>
  </div>
```



```
<style>
  :root {
    --bg-gradient-start: #0f172a;
    --bg-gradient-end: #020617;
    --card-bg: #0b1120;
    --card-border: rgba(148, 163, 184, 0.35);
    --primary: #4f46e5;
    --primary-soft: rgba(79, 70, 229, 0.18);
    --primary-hover: #6366f1;
    --text-main: #e5e7eb;
    --text-muted: #9ca3af;
    --danger: #f97373;
    --warning: #fbbf24;
    --radius-lg: 18px;
    --radius-md: 12px;
    --shadow-soft: 0 18px 45px rgba(15, 23, 42, 0.8);
    --shadow-chip: 0 0 0 1px rgba(148, 163, 184, 0.22), 0 10px 25px rgba(15, 23, 42, 0.7);
    --chip-bg: rgba(15, 23, 42, 0.85);
  }

  * {
    box-sizing: border-box;
  }

  body {
    margin: 0;
    min-height: 100vh;
    font-family: system-ui, -apple-system, BlinkMacSystemFont, "SF Pro Text",
                 "Segoe UI", sans-serif;
    color: var(--text-main);
    background: radial-gradient(circle at top left, #1e293b 0, transparent 55%),
                radial-gradient(circle at bottom right, #0f172a 0, transparent 55%),
                linear-gradient(135deg, var(--bg-gradient-start), var(--bg-gradient-end));
    padding: 32px 16px 40px;
  }

  .page-shell {
    max-width: 1080px;
    margin: 0 auto;
  }
```

Workflow

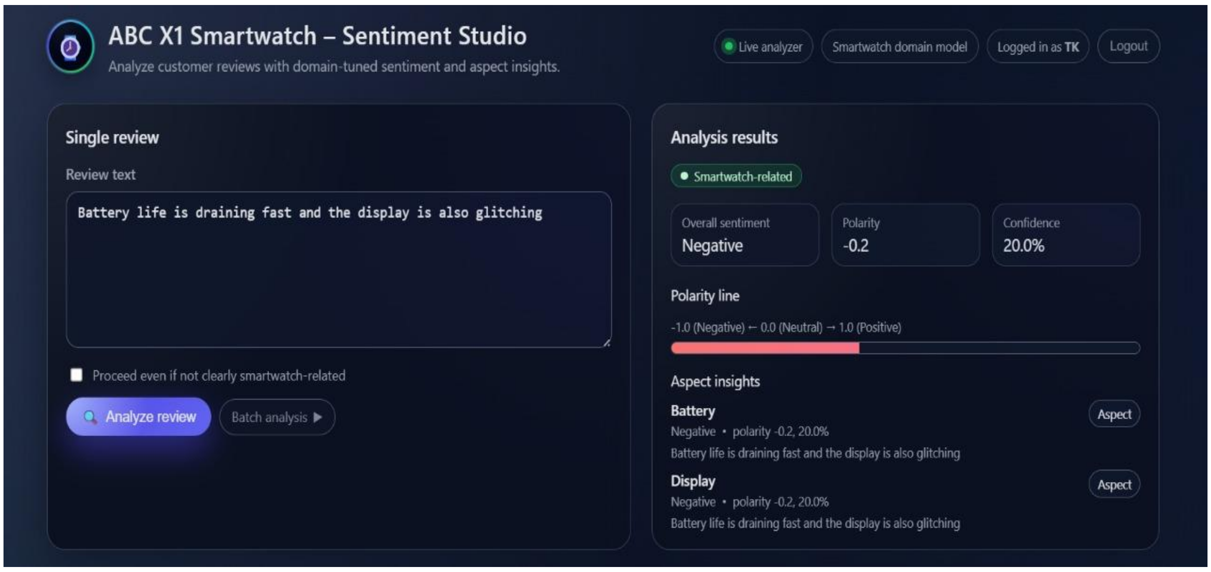
1. **Login / Signup Screen**
   - Simple form for entering username and password.
   - Once authenticated, user is redirected to the main sentiment analysis interface.



Sign in

Use your account to access the smartwatch sentiment analyzer.

2. **Single Review Interface**
   - Contains a large text area for pasting smartwatch review text.
   - Option to tick "Proceed even if not clearly smartwatch-related" for testing.
   - On submission, displays a result section with:
     - Overall sentiment label
     - Polarity value and confidence percentage
     - Colored tag or badge (e.g., green for Positive, red for Negative)
     - Aspect-wise sentiment cards.



ABC X1 Smartwatch – Sentiment Studio

Analyze customer reviews with domain-tuned sentiment and aspect insights.

Single review

Review text

```
Battery life is draining fast and the display is also glitching
```

☐ Proceed even if not clearly smartwatch-related

Analysis results

● Smartwatch-related

Overall sentiment	Polarity	Confidence
Negative	-0.2	20.0%

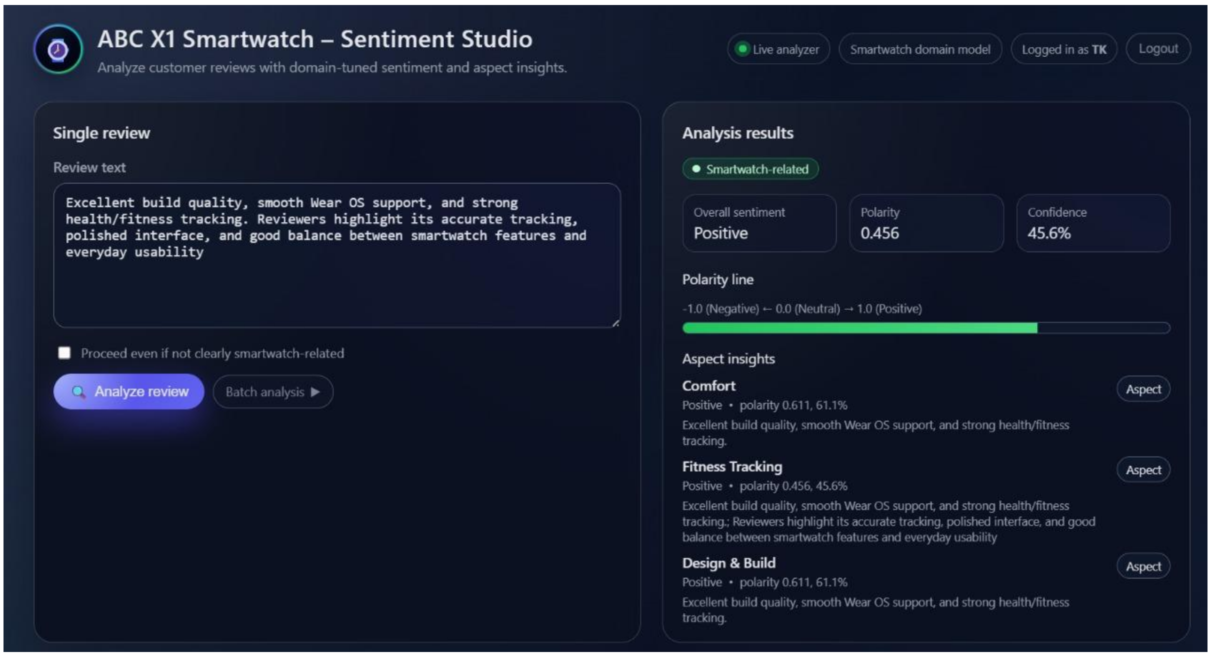
Polarity line

-1.0 (Negative) ← 0.0 (Neutral) → 1.0 (Positive)

Aspect insights

Battery
Negative • polarity -0.2, 20.0%
Battery life is draining fast and the display is also glitching

Display
Negative • polarity -0.2, 20.0%
Battery life is draining fast and the display is also glitching



ABC X1 Smartwatch – Sentiment Studio

Analyze customer reviews with domain-tuned sentiment and aspect insights.

Single review

Review text

```
Excellent build quality, smooth Wear OS support, and strong
health/fitness tracking. Reviewers highlight its accurate tracking,
polished interface, and good balance between smartwatch features and
everyday usability
```

☐ Proceed even if not clearly smartwatch-related

Analysis results

● Smartwatch-related

Overall sentiment	Polarity	Confidence
Positive	0.456	45.6%

Polarity line

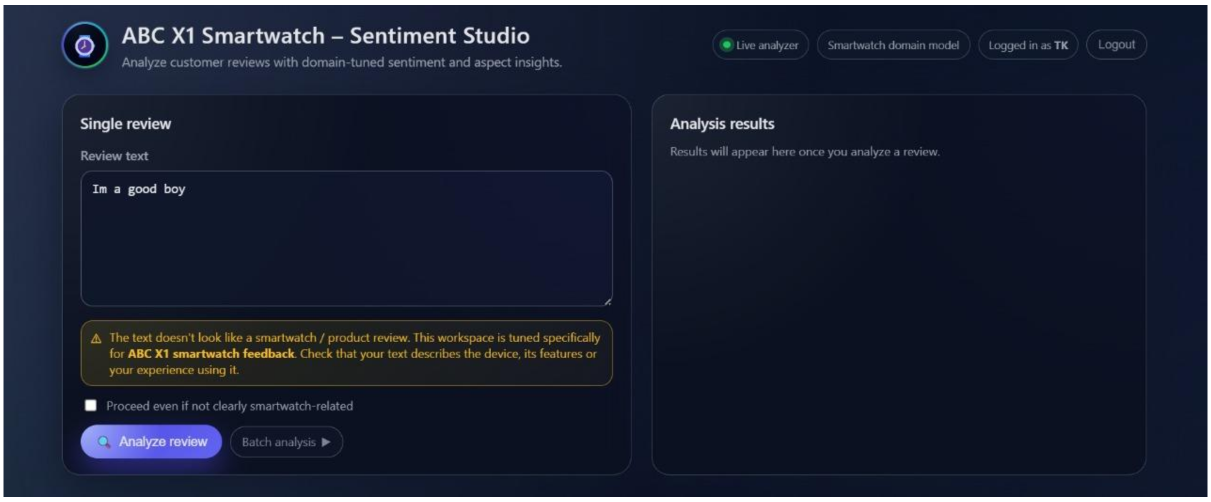
-1.0 (Negative) ← 0.0 (Neutral) → 1.0 (Positive)

Aspect insights

Comfort
Positive • polarity 0.611, 61.1%
Excellent build quality, smooth Wear OS support, and strong health/fitness tracking.

Fitness Tracking
Positive • polarity 0.456, 45.6%
Excellent build quality, smooth Wear OS support, and strong health/fitness tracking.; Reviewers highlight its accurate tracking, polished interface, and good balance between smartwatch features and everyday usability

Design & Build
Positive • polarity 0.611, 61.1%
Excellent build quality, smooth Wear OS support, and strong health/fitness tracking.



ABC X1 Smartwatch – Sentiment Studio

Analyze customer reviews with domain-tuned sentiment and aspect insights.

Single review

Review text

```
Im a good boy
```

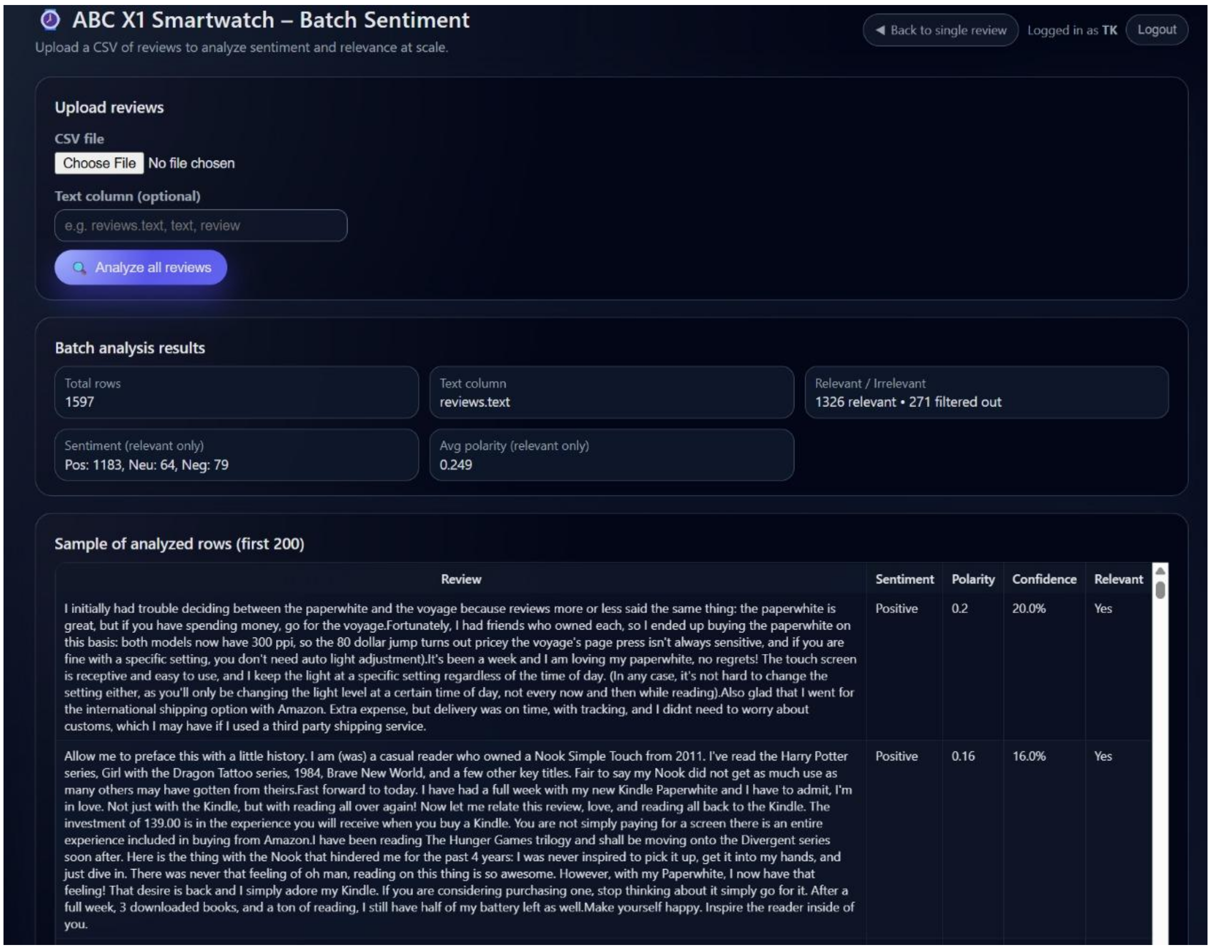
⚠ The text doesn't look like a smartwatch / product review. This workspace is tuned specifically for **ABC X1 smartwatch feedback**. Check that your text describes the device, its features or your experience using it.

☐ Proceed even if not clearly smartwatch-related

Analysis results

Results will appear here once you analyze a review.

3. **Batch Review Interface**
    - Allows CSV upload and shows the detected text column.
    - After processing, shows:
        - Summary statistics (total, relevant, positive, neutral, negative, average polarity).
        - Paginated or scrollable table of individual review predictions.


ABC X1 Smartwatch – Batch Sentiment
Upload a CSV of reviews to analyze sentiment and relevance at scale.
◀ Back to single review
Logged in as TK
Logout
Upload reviews
CSV file
Choose File
No file chosen
Text column (optional)
e.g. reviews.text, text, review
Analyze all reviews

Batch analysis results

Total rows 1597	Text column reviews.text	Relevant / Irrelevant 1326 relevant • 271 filtered out
Sentiment (relevant only) Pos: 1183, Neu: 64, Neg: 79	Avg polarity (relevant only) 0.249	

Sample of analyzed rows (first 200)

Review	Sentiment	Polarity	Confidence	Relevant
I initially had trouble deciding between the paperwhite and the voyage because reviews more or less said the same thing: the paperwhite is great, but if you have spending money, go for the voyage.Fortunately, I had friends who owned each, so I ended up buying the paperwhite on this basis: both models now have 300 ppi, so the 80 dollar jump turns out pricey the voyage's page press isn't always sensitive, and if you are fine with a specific setting, you don't need auto light adjustment).It's been a week and I am loving my paperwhite, no regrets! The touch screen is receptive and easy to use, and I keep the light at a specific setting regardless of the time of day. (In any case, it's not hard to change the setting either, as you'll only be changing the light level at a certain time of day, not every now and then while reading).Also glad that I went for the international shipping option with Amazon. Extra expense, but delivery was on time, with tracking, and I didnt need to worry about customs, which I may have if I used a third party shipping service.	Positive	0.2	20.0%	Yes
Allow me to preface this with a little history. I am (was) a casual reader who owned a Nook Simple Touch from 2011. I've read the Harry Potter series, Girl with the Dragon Tattoo series, 1984, Brave New World, and a few other key titles. Fair to say my Nook did not get as much use as many others may have gotten from theirs.Fast forward to today. I have had a full week with my new Kindle Paperwhite and I have to admit, I'm in love. Not just with the Kindle, but with reading all over again! Now let me relate this review, love, and reading all back to the Kindle. The investment of 139.00 is in the experience you will receive when you buy a Kindle. You are not simply paying for a screen there is an entire experience included in buying from Amazon.I have been reading The Hunger Games trilogy and shall be moving onto the Divergent series soon after. Here is the thing with the Nook that hindered me for the past 4 years: I was never inspired to pick it up, get it into my hands, and just dive in. There was never that feeling of oh man, reading on this thing is so awesome. However, with my Paperwhite, I now have that feeling! That desire is back and I simply adore my Kindle. If you are considering purchasing one, stop thinking about it simply go for it. After a full week, 3 downloaded books, and a ton of reading, I still have half of my battery left as well.Make yourself happy. Inspire the reader inside of you.	Positive	0.16	16.0%	Yes

4. Review Input Form

Form Components

- **User Input:**
    - **Review Text:** Text area for users to input smartwatch reviews.
    - Optional: Dropdown to select **prediction model** (Classical ML vs. Transformer-based) for advanced users.
- **Focus:**
    - The input form is streamlined to a single essential field (review text) to maintain simplicity and reduce user error.

- Mobile-responsive design ensures usability across devices.

5. Sentiment Analysis Results

Results Include:

- **Sentiment Classification:**
  - Predicted sentiment is displayed as "Positive Sentiment" or "Negative Sentiment".
- **Additional Insights (Future Integration):**
  - Could include sentiment trends for a specific smartwatch brand or product.
  - Aggregate metrics if multiple reviews are submitted.

Milestone 4: Testing & Validation

Activity 4.1: Functional Testing

Tested all core features:

Login & Authentication

- Successful signup
- Incorrect password handling
- Session-based login and logout

Single Review Analysis

- Valid smartwatch reviews
- Non-smartwatch reviews (correctly flagged)
- Mixed reviews
- Very short reviews
- Emoji-containing reviews

Domain Rule Validation

Tested the rule engine with various triggers:

- Battery draining → Negative
- Good battery life → Positive
- Screen scratch → Negative
- Comfortable strap → Positive

All rules worked correctly.

Activity 4.2: Batch Processing Testing

- CSV with correct review column
- CSV with multiple text columns
- CSV with missing review text
- CSV with irrelevant reviews
- Large CSV file to check performance

System correctly:

- Identified text column
- Analyzed rows

- Produced summary
- Ignored empty entries

Activity 4.3: API Testing

Using Postman:

- Valid JSON input → correct sentiment output
- Missing fields → error message
- Incorrect MIME types → handled safely
- Batch API validated with uploaded CSV

APIs performed consistently.

Activity 4.4: User Interface Testing

Verified:

- Button actions
- Page navigation
- Templates rendering properly
- Scrollable tables working
- Color-coded sentiment labels

Future Implementations

Potential improvements and extensions of the Smartwatch Sentiment Analyzer include:

- **Integration with E-Commerce Platforms**
    - Develop APIs that can be integrated directly into seller dashboards to continuously monitor smartwatch reviews in real time.
- **Using Transformer Model in the Web App**
    - Replace or complement the TextBlob baseline with the fine-tuned DistilBERT model for more accurate predictions in the live web interface.
- **Aspect-Based Transformer Models**
    - Train specialized models that directly predict sentiment for each aspect (battery, display, comfort, etc.) instead of overall sentiment only.
- **Multilingual Support**
    - Extend the system to handle reviews written in different languages using multilingual transformer models.
- **Visualization Dashboard**
    - Build an analytics dashboard showing trends over time, word clouds, and heatmaps of sentiment across different smartwatch models.
- **Continuous Learning**
    - Periodically retrain models with newly collected reviews so that the system adapts to new products, slang, and user preferences.

Conclusion

The Smartwatch Sentiment Analyzer project successfully demonstrates how machine learning and natural language processing can be applied to understand customer opinions about wearable devices.

Starting from raw smartwatch reviews, the project builds a complete pipeline that includes data cleaning, exploratory analysis, model training with both classical ML and transformer-based approaches, and a practical Flask web application for interactive analysis.

The comparison between TF–IDF Logistic Regression and DistilBERT clearly shows the benefits of contextual language models in handling informal text, nuanced opinions, and domain-specific vocabulary. The addition of a TextBlob-based baseline with domain rules and aspect-wise analysis makes the system interpretable and useful for real product teams.

Overall, the final system is a robust and extensible platform for smartwatch sentiment analysis. It not only provides accurate predictions but also gives actionable insights that can guide product design, marketing strategies, and customer support improvements.