BTP PROJECT PLAN

1) We have 2k labelled rows and 14k unlabelled Rows. In labelled dataset we have integrity label(-1,0,1) and language is English and Hinglish. Also review time is also associated

2) In unlabelled Dataset we have 14k rows. Languages are 6.

3) We can do topic labelling of this dataset. The research paper tells about the domain of topics in the dataset we can tell gpt to label it

4) We will carry out the research on the smaller 2k dataset labelled first to test our model. If it works fine we can then proceed to larger dataset. We will train classifiers on integrity and topic to label the larger dataset.

5) How are we gonna label these comments topic wise

->By Few-shot learning refers to teaching a language model **a new task** by showing it just a few examples **within the prompt** — no fine-tuning required.

In your case:

> You're teaching the LLM how to assign a topic to a civic comment by **giving it examples of comment → topic mappings** in the prompt itself.

This works well because **LLMs like GPT-4, Claude, and Gemini** have been pretrained on massive text corpora and can **generalize from small patterns**.

->We can setup a prompt like
You are a civic analyst. Your job is to assign a topic from a predefined list to each citizen comment.

Possible topics:
[Healthcare, Education, Urban Policy, Climate Change, Law & Order, Water Management, Environment, Transport, Digital India, Rural Development]

Here are some examples:

Comment: "The sewage problem in my colony needs urgent attention."
Topic: Urban Policy

Comment: "Children in my village lack access to basic schooling."
Topic: Education

Comment: "I appreciate the steps taken to reduce pollution in Delhi."
Topic: Environment

Now assign a topic to the following comment:

Comment: "We need more mobile medical vans in rural areas."

Topic:

-> We can then manually review these labelled comments

6) It will basically be a content based recommendation system. Because we dont have as such user and entity relations. We will generate the user embedding using the comments that he gives. We will calculate the similarity between all the users using cousine similarity.We will have top N users and the comments that they have posted we will recommend it to the user.This is the broad pipeline.

7)Now how are we gonna generate User Embeddings? First we will need to generate the comment embeddings. How we are going to do it.

7) We will use the concept of supervised training. We will send the comment text through an embedding layer to generate the embeddings now to make these embeddings aware of the integrity and topic of the comment we will use supervised learning. The comment embedding will pe passed through a neural network that will predict the integrity label and comment topic. We will then use logit loss and it will backpropogate to the comment embedding to optimize it. This is how we are going to get contextualized embedding aware of the integrity and the topic.

8) Now we have the comment embedding how are we gonna generate the user embedding? We will take the comment embeddings of the comments user have given and aggregate them to form user embedding. The aggregation will be done using time decay. We will use the review time in the dataset and multiple a time decay formula, This is done so that the recently commented embedding are more time aware. Recent comments will have more impact on the user embedding. We take a mean of the time decayed comments to form user embedding.

9) Now we have the user embedding. We will generate a similarity matrix for users where mat[i][j] will tell the cosine similarity of user i with user j. We will find top N users and collect the reviews of those users and recommend to that user.

10) One thing we noticed this that the user embedding of legitimate users will be together in the vector space and non legitimate users will be together. When we apply this cosine similarity with users legitimate users will be similar to legitimate and non legitimate with non legitimate. So legitimate users will get recommendation of legitimate and non legitimate of non legitimate. This will be a problem as non legitimate users will get recommendations of non legitimate posts and this will not be the essence of our reliability recommender system.

How to handle this issue

11) For this we need to label the users as legitimate and non legitimate. For legitimate we can use the conventional user-user cosine similarity. But for non legitimate we will recommend by getting the topics on which they have commented. And giving them the recommendation of comments of the matched topics which are legitimate.Basically taking a topic and finding the similar comment which is legitimate. In this way non legitimate will get recommendation of legitimate.

12) How are we gonna label these users. They are not done initially. We will label a portion of users by calculating the mean of integrity of their comments. Let us say a user has done 5 comments 4 legitimate 1 non legitimate overall 0.8 score which is labelled as 1 like this. So we have the labelling of users now we will train a classifier that will take user embedding and label by integrity. For training of this model

we will take the embedding of pseudo labelled users and train our model.

13) Now this recommendation system will be dynamic if user comments then its embedding will be revised on the basis of this comment. And its recommendation will alson be accordingly changed. We will pass the new comment through the supervised learning model to generate its embedding and will integrate it to its user embedding.

14) Also the user can select a particular comment as harmful or dislike. That will be kept in discarded list. When we recommend the user again, these recommendations will pe passed through a filtering layer where cosine similarity with these discarded comments will be calculated and the ones having high similarity will be discarded.

15) How we will judge the quality of recommendations

In our pipeline we have 3 4 models to train Topic classifier, Integrity classifier, Comment embedding generator, User Integrity calculator.

->For normal classifier like Integrity, Topic, User integrity we can use simple ROC curve, Confusion matrix

->But for Comment Embedding,

🔹 1. Semantic Structure & Discrimination

✅ A. Cosine Similarity Tests

- Measure average cosine similarity between:

- Same-topic, same-integrity comment pairs → should be **high**
- Different-topic or opposing-integrity pairs → should be **low**

Useful for sanity-checking that embeddings reflect actual content and labels.

✅ B. Clustering Metrics

Apply KMeans, DBSCAN, or HDBSCAN on the comment embeddings and measure:

Metric	Purpose	Use
Silhouette Score	Measures how distinct the clusters are	High = better separation
Davies-Bouldin Index	Lower is better	Measures cluster compactness
Calinski-Harabasz Index	Higher = better	Ratio of inter-cluster to intra-cluster distance
Topic Purity	% of dominant topic per cluster	Check if topic info is preserved

---

🔹 2. Visual Separation & Interpretation

✅ C. t-SNE / UMAP Visualization

- Reduce embeddings to 2D

- Color by topic or integrity label

> You should see **tight clusters** of same-topic or same-trust comments.

🔹 3. Downstream Utility

If the embedding is good, downstream tasks should be easy.

✅ D. Classification from Frozen Embeddings

Train lightweight models (like Logistic Regression or MLP) on frozen embeddings to predict:

- Topic
- Integrity

> **High accuracy = good embedding representation.**

->For User embedding

🔹 1. Semantic & Structural Evaluation

✅ A. User Clustering Quality

Apply clustering on user embeddings and analyze:

Metric	Purpose
Silhouette Score	Separation of user groups (by topic or behavior)
Topic/User Group Purity	% of users in each cluster who share topic interest or trust level
Inter/Intra Cluster Distance	Distribution spread check

✅ B. t-SNE / UMAP Visualization

- Reduce to 2D
- Color-code by:
    - Primary topic of user's comments
    - User trust label (genuine/ingenuine)

Visually verify that similar users group together.

🔹 2. Embedding Consistency & Stability

✅ C. Cosine Similarity Among Same-Type Users

- Average cosine similarity between:
    - High-trust users ↔ high-trust users → should be **high**
    - High-trust ↔ low-trust → should be **low**
    - Same-topic users → **moderate to high**

✅ D. Time-Based Drift

Track a user's embedding after each new comment:

text

CopyEdit

```
U_t+1 = 0.8 × U_t + 0.2 × new_comment_embedding
```

- Drift should be **gradual**, not erratic
- Use:
    - **Cosine distance between U_t and U_t+1**
    - Or **KL divergence** over user embedding distribution across time

---

🔹 3. Downstream Evaluation: Classification

✅ E. User Integrity Prediction Accuracy

Train a classifier:

- Input: `UserEmbedding`
- Output: `Genuine` / `Not genuine`

Metrics:

- Accuracy
- Macro F1
- AUROC

> A good user embedding should **naturally separate genuine and ingenuine users**.

✅ F. Personalized Recommendation Performance

Use `UserEmbedding` for cosine similarity to other users:

- Retrieve top-k similar users
- Recommend their high-integrity comments
- Evaluate recommendation metrics:
    - Precision@k

- nDCG
- Trusted@k (how many from high-integrity users?)

16) Explainability of these recommendations to user why they got this comment recommended

🧠 Explainability in the Integrity-Aware Recommender System

Your recommendations aren't just based on matching — they're **ethically filtered** (via integrity scores), **personalized** (via embeddings), and **topically relevant**.

Explainability should:

1. **Justify why a recommendation was shown**
2. **Build trust in the system**
3. **Expose key metadata: similarity, topic, trust, recency**

🔹 1. Explainability for Legitimate (Trusted) Users

Personalization and Integrity-Based Reasoning:

> *"You received this recommendation because:*
>
> - *It's from a **trusted user** whose comment profile is **87% similar** to yours.*

- *It matches your interest in **#EducationPolicy**.*
- *It shares **91% semantic similarity** with your recent comment.*
- *It was posted **2 days ago**.”*

✅ Metadata Elements to Include:

Feature	Source
Source user integrity	Integrity classifier (trusted/untrusted)
User similarity	Cosine similarity between user embeddings
Topic match	Topic classification result
Comment similarity	Cosine similarity between comment embeddings
Timestamp	Comment's `review_time`

✅ Display Format (Frontend-friendly JSON):

json

CopyEdit

```
{
  "explanation": {
    "source_user_integrity": "trusted",
    "user_similarity": "87%",
    "comment_similarity": "91%",
    "topic": "#EducationPolicy",
    "review_time": "2025-07-10"
  }
}
```

🔴 2. Explainability for Non-Legitimate (Untrusted) Users

These users are **not personalized** via embeddings to avoid misuse. We use a **topic-based fallback**.

> *“This is a reliable opinion on **#Healthcare**, a topic you’ve recently posted about.*
> *It comes from a **verified contributor** with a high trust score and shares **83% similarity** with your recent comment.”*

✅ Simplified Metadata:

- **Topic match**
- **Comment similarity (optional)**
- **Source integrity**

json

CopyEdit

```
{
  "explanation": {
    "topic": "#Healthcare",
    "comment_similarity": "83%",
    "source_user_integrity": "trusted"
  }
}
```

Now this is the complete pipeline

If it works then we will switch to larger dataset. We will first translate them to english. Then we will train classifiers for integrity and topic and label the large dataset. And run the same things on it.

There is one challenge of Topic labelling. Class size!!!

We can undersample the majority class or discard the minority class.
Or by using class weights