Phase	Task	Date	Decisions & Rationale	Challenges & Solutions	Implementation
1:Data Collection Research and Assess	**Collect comments from YouTube using API**	**21 September 2025**	Decision: Use YouTube Data API to automatically collect comments. Rationale: Efficient, reliable, and supports keyword filtering.	Challenges: API key required; some videos had comments disabled. Solutions: Used valid API key and try/except blocks to skip problematic videos	Implementation: Python with googleapiclient; fetched video IDs, retrieved comments for each video with engagement metrics, and saved to CSV using pandas.
	Collect tweets from Twitter using API	**21 September 2025**	Decision: Use Twitter API (Tweepy) to collect recent tweets with relevant keywords. Rationale: Provides structured access to tweets with engagement metrics and filtering options.	Challenges: Need to collect relevant tweets and handle API limits. Solutions: Applied keyword filtering, limited results per request, excluded retweets, and collected engagement metrics.	Implementation: Python with tweepy.Client, search_recent_tweets(), extracted tweet text and metrics, stored in pandas.DataFrame and saved with to_csv().
	Collect comments from Reddit using API	**22 September 2025**	Used praw to collect comments automatically as it provides structured access and engagement data.	API limits and search errors → set post limits, applied time filter, and used try/except to skip failed requests.	Used praw.Reddit + subreddit.search() to fetch comments, stored results in a pandas.DataFrame, and exported to CSV.
	Clean the collected comments	**22 September 2025**	Decision: Apply text cleaning to improve data quality. Rationale: Cleaning ensures reliable and accurate analysis by removing noise and irrelevant information	Challenge: The dataset contained duplicate comments, empty rows, URLs, mentions, emojis, punctuation, and stopwords that could affect analysis. Solution: Applied a series of preprocessing steps to remove duplicates, empty rows, URLs, mentions, emojis, punctuation, stopwords, and reset the DataFrame index.	Used the following Python functions/methods: drop_duplicates() dropna() reset_index() re.sub() for URLs and mentions re.compile().sub() for emojis str.translate() for punctuation stopwords.words() and list comprehension for stopwords removal
	Measure and visualize the proportion of hate speech per platform.	**24 September 2025**	Decided to calculate hate speech proportions using normalized labels (1 = hate, 0 = non-hate) and visualize results with a bar chart. This provides a clear comparison across platforms.	Some datasets had different label formats (text vs numbers). Solution: Standardized labels by mapping text (e.g., "toxic", "hate") to numeric values and dropping invalid rows.	Normalized label column to numeric (0/1). Filtered valid rows only. Calculated hate speech proportion for each platform. Visualized results using matplotlib bar plot with percentages on top.

	Visualize engagement distribution for hate vs non-hate content across platforms.	**24 September 2025**	Use boxplots to compare engagement levels (likes, replies, etc.) between hate and non-hate posts. Boxplots clearly show medians, ranges, and spread, making comparisons easier.	Labels and engagement columns had inconsistent formats → Solution: Standardize labels using to_binary_label() and convert engagement to numeric. Some rows missing data → Solution: Filter out invalid rows before plotting.	Convert label column to numeric (0/1) using to_binary_label. Filter valid rows with both label and engagement. Separate data by platform and label. Plot boxplots with matplotlib, showing engagement distribution for each label per platform.