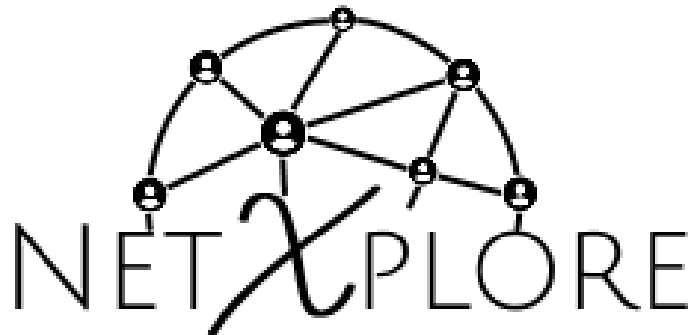




Software Engineering B.Sc. Final Project

Authors:

Tslil Aharon 318320694,
Noy Raichman 318734886,
Shira Dagan- Roberts 315550194

Supervisor:

Dr. Dvir Ross

A thesis submitted in partial fulfillment of the requirements for the degree of BSc. in Software Engineering

Abstract

NetXplore is an advanced analytical platform designed to process, compare, and visualize social discourse from heterogeneous sources, primarily WhatsApp conversations and Wikipedia talk pages. The system implements an end-to-end workflow, from data ingestion and anonymization to graph construction and interactive analysis, combining community detection, network metrics, and the novel Distance Weighted Ranking method to reveal structural and behavioral insights hidden in raw text.

A central feature of NetXplore is its interactive graph visualization, where participants are represented as nodes and interactions as edges. This enables researchers to dynamically explore relationships, highlight key actors, and identify community structures and communication patterns not visible through manual reading.

Beyond visualization, the platform offers filtering and comparison tools, allowing analysis across time ranges, keywords, and engagement levels, as well as side-by-side dataset comparisons. By bridging the gap between linear WhatsApp logs and hierarchical Wikipedia threads, NetXplore provides a unified, researcher-friendly interface that supports both qualitative exploration and quantitative measurement, ultimately transforming raw conversations into meaningful, visually grounded insights.

Acknowledgements

We would like to express our sincere gratitude to Dr. Dvir Rus for his guidance, encouragement, and professional insights, which were of decisive importance in shaping the direction of this project. His dedication, availability, and constructive feedback accompanied us at every stage and greatly enriched both the process and the outcome.

We wish to thank Prof. Avivit Levi, Head of the Software Engineering Department, for her guidance, support, and continuous assistance throughout the project, as well as for connecting us with Assaf Shapira, whose contribution was highly significant to the project's success.

We also extend our thanks to Assaf Shapira for his contribution to the system's validation, for connecting us with the network research community, and for the valuable comments and insights he shared from his academic expertise, which helped improve the system and provided us with a broader perspective of the field.

Finally, we would like to thank the Faculty of Engineering at Shenkar College for the academic framework and resources it provided, which enabled the successful development of the NetXplore project..

Table of Contents

1. INTRODUCTION	7
1.1 Problem Description	7
1.2 Motivation	8
1.3 Goals	8
1.4 System Overview	8
1.5 Usage Scenarios	9
Scenario 1 – Analyzing a Topic Focused Conversation	9
Scenario 2 – Analyzing a Wikipedia Talk-Page Discussion	10
Scenario 3 – Comparing Communication Patterns in Two WhatsApp Groups	11
1.6 Audience	12
1.7 Glossary	12
1.7.1. NetXplore Terminology	12
1.7.2. Generic Terminology	13
2. Literature Survey	13
2.1 Problem Survey	13
2.2. Solution Survey	14
2.3 Interactive Graph Analysis	14
2.4 Platform Design and User Interaction	15
3. Technological Survey	15
3.1. Problem Survey	15
3.2. Solution Survey	16
3.3. Discussion and Conclusion	17
4. Requirements and Specification	18
4.1 Functional Requirements	18
4.1.1 Researcher Platform	18
4.1.2 Data Processing Pipeline	19
4.3 Specification: The Scope of NetXplore	20
4.4 The NetXplore approach	20
5. The Architecture	21
5.1 System Architecture	22
5.2 NetXplore Data	22
5.2.1 Researcher Data	25
5.2.2 Initial Start-up Data	27
5.2.3 Data Collection	28
5.3 NetXplore Process	29
5.3.1 Graph Construction and Metrics	30
5.3.2 NetXplore Network Analysis Process	31
5.3.3 Filtering, Highlighting, and Comparison Process	32
5.4 The life-cycle of a system	34
5.5. Roles	36

6. SYSTEM DESIGN	37
6.1 Use Cases	37
6.2 Class Diagram	38
6.3 Data Components	39
6.4 Interactions	40
6.4.1 Activity diagram	40
6.4.2 Sequence Diagram	41
6.5 Software Architecture Pattern	44
6.5.1 Three-tier: Data, Logic and Presentation tiers	44
6.5.1 MVC: Model, View, Controller Structure	46
7. IMPLEMENTATION	48
7.1 Algorithm Design	48
7.1.1 Description of Algorithmic Components	48
7.1.2 Distance Weighted Ranking (DWR) Algorithm	49
7.2 Interfaces	54
7.2.1 NetXplore Welcome Page.	54
7.2.2 Login Page	55
7.2.3 Home Dashboard	56
7.2.4 Research History	57
7.2.5 Platform Selection Page	57
7.2.6 Research Setup Wikipedia	58
7.2.7 Discussion page	59
7.2.8 Research Setup WhatsApp	60
7.2.9 Configuration Page	61
7.2.10 Time Frame Page	63
7.2.11 User Filters Page	64
7.2.12 Network Analysis Page	65
7.2.13 Network Data Table	66
7.2.14 Visualization Customization	67
7.2.15 Comparison Page	68
7.2.16 Research Report Page	69
7.2.17 Report Generation Popup	70
7.2.18 Save Research Pop-Up	71
7.3 Development environment	71
7.4 Programming languages	72
7.5 Risk Management	73
7.6 Exceptions Management	74
7.7 Versions Control	74
7.8 Project Management	75
7.9 Code	78

8. System Validation	78
8.1 Functional Validation	79
8.2 Usability Validation	81
8.3 Algorithmic Validation	82
9. SUMMARY, EVALUATION, CONCLUSIONS AND FUTURE WORK	84
9.1 Summary	84
9.2 Evaluation and Conclusions	84
9.3 Future Work	84
10.REFERENCES	86

Table of Figures

Figure 3.2 – Selected Technologies for NetXplore	17
Figure 5.1 - System Architecture	22
Figure 5.3.2 – Social network analysis process of NetXplore	32
Figure 5.2.3 – Comparison Process of NetXplore	33
Figure 5.6 – System life-cycle of NetXplore	35
Figure 6.2 – Class diagram of NetXplore	38
Figure 6.3 – Data Components EDR diagram of NetXplore	39
Figure 6.4.1 - Activity Architecture of NetXplore	40
Figure 6.4.2-1 – Sequence diagram of NetXplore data upload and graph construction	41
Figure 6.4.2-2 – Sequence diagram of NetXplore filter and update parameters	42
Figure 6.4.2-3 – Sequence diagram of NetXplore dataset comparison (A & B)	42
Figure 6.4.2-4 – Sequence diagram of NetXplore export report (CSV/PDF)	43
Figure 6.5.1 - Three-tier Diagram	44
Figure 6.5.2 - Model, View, Controller Structure	46
Figure 7.1.2 - 1 – Raw WhatsApp Conversation	49
Figure 7.1.2 - 2 – DWR Workflow Diagram	51
Figure 7.2.1- Welcome Screen Page	54
Figure 7.2.2 - Sign In Screen Page	55
Figure 7.2.3- Home Dashboard Screen Page	56
Figure 7.2.4- Research History Screen Page	57
Figure 7.2.5- Platform Selection Screen Page	57
Figure 7.2.6- Research Setup Wikipedia Screen Page	58
Figure 7.2.7- Discussion Screen Page	59
Figure 7.2.8- Research Setup WhatsApp Screen Page	60
Figure 7.2.9 - Configuration Page Screen Page	61
Figure 7.2.10 - Time Frame Screen Page	63
Figure 7.2.11 - User Filters Page	64
Figure 7.2.12 - Network Analysis Page	65
Figure 7.2.13 -Network Data Table	66
Figure 7.2.14 - Visualization Customization	67
Figure 7.2.15 - Comparison Page	68
Figure 7.2.16 - Research Report Page	69
Figure 7.2.17 - Report Generation Popup	70
Figure 7.2.4 - Save Research Pop-Up	71

1. INTRODUCTION

The rapid expansion of online communication platforms has produced unprecedented volumes of conversational data, spanning private group messaging applications to public collaborative forums. Extracting meaningful insights from such data presents considerable challenges, particularly given the structural diversity of these platforms. WhatsApp conversations follow a linear, chronological flow, whereas Wikipedia talk pages are inherently hierarchical, consisting of nested threads and layered sub-discussions.

Conventional analytical methods often struggle to accommodate these differences, leading to incomplete representations of relationships, communication patterns and community structures embedded within the data. As a result, researchers require analytical tools that can process heterogeneous conversational formats while presenting the outcomes in an intuitive, interactive and research-oriented manner.

NetXplore was developed in response to this need. The platform ingests raw conversational data from WhatsApp and Wikipedia, offers researchers the option to anonymise participant identities, structures the data into a uniform format and transforms it into interactive social network graphs. In these graphs, participants are represented as nodes and their interactions as edges, enabling researchers to dynamically explore connections, detect communities and identify patterns of communication that remain hidden in text-based analysis. By integrating advanced filtering options, structural network metrics and community detection algorithms, NetXplore bridges the gap between fragmented conversation logs and actionable, visually driven insights, offering a powerful tool for both qualitative and quantitative research.

1.1 Problem Description

The proliferation of online communication platforms has resulted in the accumulation of vast amounts of conversational data, containing rich social, behavioral, and structural information. Despite this potential, the raw format of such data is inherently unstructured, making it challenging to systematically uncover the underlying relationships between participants and the dynamics of their interactions.

Platforms such as WhatsApp and Wikipedia exemplify this challenge. WhatsApp conversations are represented as long, linear message streams, whereas Wikipedia talk pages follow a hierarchical structure composed of nested threads and sub-discussions. Although these formats are optimized for human readability, they are poorly suited for computational analysis. As a result, key features such as interaction patterns, emergent communities, and cross-platform comparisons remain obscured.

In the absence of specialized processing and visualization methods, researchers are forced to rely on manual reading and qualitative inspection of conversations. This approach is not only time-consuming and prone to subjective bias but also fails to scale to the size and complexity of modern datasets. Moreover, it severely limits the capacity for quantitative, comparative, and reproducible analysis across different groups, topics, or platforms.

1.2 Motivation

The increasing relevance of social network analysis in disciplines such as sociology, communication studies, and digital community research has intensified the need for tools capable of transforming unstructured conversational data into actionable insights. Although platforms such as WhatsApp and Wikipedia preserve extensive records of rich discourse, their native representations, whether in the form of linear message streams or hierarchical discussion threads, are optimized for human readability rather than computational analysis. As a result, essential features such as interaction dynamics, participant relationships, and emergent community structures often remain hidden.

To address this limitation, researchers require solutions that process heterogeneous conversational formats and present results through clear, intuitive, and interactive interfaces. Visualizing discussions as network graphs offers a powerful approach, enabling analysts to identify central participants, uncover influential connections, and compare communication dynamics across groups and contexts.

NetXplore was developed with this objective in mind, bridging the gap between raw conversation logs and structured network representations. Through efficient preprocessing, anonymization, and advanced visualization, the system provides a robust framework that supports both qualitative exploration and quantitative measurement. This dual capacity positions NetXplore as a valuable resource for academic research, organizational analysis, and investigative studies, enabling a deeper understanding of online social interaction.

1.3 Goals

The goal of NetXplore is to transform conversational data from WhatsApp and Wikipedia into clear and interpretable network insights. The system converts raw chats and talk pages into social network graphs that represent participants and their interactions, enabling researchers to identify structures, key actors, and emerging communities.

Through interactive visualization, filtering options, and comparative analysis, NetXplore facilitates focused exploration of discussions while integrating core network metrics that support both qualitative interpretation and quantitative reasoning. Privacy is preserved through built-in anonymization, and the platform further assists research reporting by allowing selected views, filters, and metrics to be saved and reused.

1.4 System Overview

NetXplore is a web-based research platform designed to transform conversational data from WhatsApp and Wikipedia into interactive social network graphs that support visual and data-driven analysis. The system provides an end-to-end workflow that begins with importing raw datasets in the form of WhatsApp TXT exports or Wikipedia talk pages. These datasets are parsed and normalized to extract participants, timestamps, and reply relations, ensuring consistency across different sources.

To account for the structural differences between platforms, NetXplore employs specialized processors. The WhatsApp processor interprets conversations as a linear chronological flow, while the Wikipedia processor reconstructs the hierarchical structure of threaded replies, allowing the system to accurately infer links between participants. Once processed, the data is transformed into network graphs in which participants are represented as nodes and their interactions as edges. These graphs can be directed or undirected, and weighted according to interaction frequency, sequence, or the Distance Weighted Ranking method.

The platform integrates analysis services that compute core network metrics, identify communities, and prepare the data for visual exploration and filtering. All processed data, graph representations, and saved analyses are stored in a PostgreSQL database managed through Xata, enabling efficient retrieval and long-term persistence. The frontend application, developed with React, provides researchers with an interactive environment to explore the graphs, apply filters in real time, highlight influential participants, and compare multiple datasets side by side.

In addition to interactive exploration, NetXplore supports export and reporting functionalities that allow users to save selected filters, visualizations, and analytical results for documentation and future research use. This combination of automated preprocessing, tailored graph construction, metric computation, and interactive visualization creates a comprehensive platform that bridges the gap between raw conversational logs and meaningful, interpretable social network insights.

1.5 Usage Scenarios

NetXplore is designed to support a variety of research and analysis contexts where understanding the structure and dynamics of conversations is essential. The following scenarios illustrate how the system can be applied:

Scenario 1 – Analyzing a Topic Focused Conversation

Background:

A media researcher is investigating how participants in a WhatsApp group discuss the topic of elections over the course of a specific week. She wants to focus exclusively on messages relevant to the topic and preserve the analysis for future reference.

Objective:

To build a filtered communication network around a specific topic and time frame, extract insights on participant roles and community structure, and save the resulting research within the platform for later access.

Process Overview:

1. Uploading the conversation and setting filters: The researcher exports the WhatsApp chat as a .txt file, uploads it to NetXplore, and applies filters such as date range, keywords (“elections,” “voting”), minimum message length, and participant activity to narrow the analysis to relevant content.

2. Running the analysis: She initiates the analysis. NetXplore constructs a communication graph from all messages that meet the filter criteria, calculates centrality metrics (Degree, Betweenness, Closeness), and runs a community-detection algorithm to identify clusters of participants.
3. Interpreting the results: The researcher explores the interactive network: she identifies who leads the discussion, how participants are grouped into communities, and what relationships exist between those groups. Using the metrics and community assignments, she draws conclusions about key influencers and the flow of information around the topic.
4. Saving the research: Satisfied with the insights, she selects the “Save Research” option. NetXplore stores all elements of the analysis including the research title and description, the selected filters and algorithm, and the network data in her account. This creates a persistent record in the research history.
5. Accessing history: At any time, she can open the “History” page to view all previously saved analyses.

Expected Outcome:

The researcher obtains a focused communication network for the chosen topic and timeframe, understands who drives the discussion and how the conversation is structured, and saves the study in NetXplore’s history. This enables her to revisit and analyze the findings later or to build upon them in future research.

Scenario 2 – Analyzing a Wikipedia Talk-Page Discussion

Background:

A medical researcher, aiming to explore how online debates reflect public perspectives on major health issues, chooses to analyze a section from the Hebrew *Wikipedia talk page for the COVID-19 pandemic*. She focuses on one of the most active threads to understand who the central contributors are, how participants interact, and what communication patterns shape the discussion.

Objective:

Transform a linear talk-page exchange into a network map of interactions, spotlight the most influential editors using centrality measures, and delineate natural groups of participants.

Process Overview:

1. Choosing the section and scoping the data: Upon logging into NetXplore in Wikipedia mode, she reviews the list of talk-page sections each tagged with its comment count and searchable by title. Selecting one that’s either highly active or thematically relevant, she trims the dataset by specifying date ranges or keywords so that only pertinent comments are included.
2. Creating the interaction map: NetXplore processes the chosen discussion and builds a graph of editor interactions. It calculates centrality metrics such as degree, betweenness, and closeness. A community-detection algorithm is then applied to identify clusters of editors.

3. Deriving insights: With the resulting visual, she examines which editors play central roles, how contributors are grouped, and what the interaction patterns suggest about the debate's dynamics. This allows her to identify leading voices and mediator roles.
4. Documenting the findings: She assigns a title and description to her study and saves it. NetXplore records the filtered data, computed metrics and parameters under her user account, ensuring the analysis can be revisited.
5. Reviewing past analyses: Later, she navigates to the history section to view this and other saved studies, enabling comparisons across multiple discussions or export of the data for further use.

Expected Outcome:

The researcher achieves a rich visualization of the selected Wikipedia talk-page section, learning which editors wield the most influence and how the conversation unfolds. By saving the analysis in NetXplore, she can return to it for deeper exploration or to inform subsequent projects without having to recreate the network from scratch.

Scenario 3 – Comparing Communication Patterns in Two WhatsApp Groups

Background:

A communications researcher aims to understand how conversation dynamics differ between two distinct WhatsApp groups: one made up of project team members and another consisting of a customer community. The goal is to see who leads the discussions in each group and how the structures of interaction and participation compare.

Objective:

Analyze and contrast two separate WhatsApp chats to uncover differences in network structures identifying who the key participants are, how communities are partitioned, and what the interaction levels look like while also noting any similarities.

Process Overview:

1. Collecting and uploading data: The researcher exports the message histories from both WhatsApp groups and uploads them to NetXplore. During the upload, she ensures names are anonymized and uses the same time frame for both files to establish a common basis for comparison.
2. Applying tailored filters: For each group, she sets relevant filters such as shared keywords, a minimum message threshold per participant, or filtering only active users to ensure the comparison focuses on similar topics and participation levels.
3. Constructing networks and computing metrics: NetXplore generates a separate communication graph for each file, calculates centrality values for every participant, and partitions the networks into internal communities using a community-detection algorithm.
4. Using the comparison tool: Once both networks are ready, the researcher employs NetXplore's comparison function to spot notable differences for example, which participants appear in both groups, how the centrality scores of key contributors differ,

or what common links exist across the two networks. The interface makes it easy to compare the structures and activity of the two groups.

5. Preserving and documenting the findings: She assigns a title to each analysis and to the comparison result, then saves them. NetXplore stores all the data and settings used for the analysis, enabling her to revisit the comparison, probe additional contrasts, or share the insights with others.

Expected Outcome:

The researcher obtains a clear picture of the conversation patterns in each group, learns which participants are dominant in each context, how the communities are organized, and what the significant differences are between them. The findings are saved in NetXplore, allowing her to return to the comparison later or conduct further analyses based on the same data.

1.6 Audience

NetXplore is intended primarily for researchers in the domains of social network analysis, sociology, communication studies, and digital community research. The platform is designed to support scholars who aim to explore the structural and behavioral aspects of online conversations by transforming raw data into interpretable visualizations and measurable insights. Through its combination of automated processing, anonymization, and advanced graph-based analytics, NetXplore enables both qualitative interpretation and quantitative reasoning, serving as a valuable tool for academic and applied research.

1.7 Glossary

The following terms are divided into two categories: NetXplore-specific terminology and generic terminology used in the field of social network analysis.

1.7.1. NetXplore Terminology

- **Distance Weighted Ranking (DWR)** – A custom algorithm developed within NetXplore that ranks participants according to the weighted influence of preceding messages in the conversational flow.
- **WhatsApp TXT Export** – A text file generated by WhatsApp containing a chronological record of messages exchanged in a chat.
- **Wikipedia Talk Page** – A discussion space on Wikipedia where contributors coordinate edits and deliberate article content.
- **Graph Visualization** – The interactive rendering of conversational networks in NetXplore, allowing zooming, filtering, and highlighting of participants or structures.
- **Graph Comparison** – The side-by-side display and analysis of multiple conversation graphs, enabling the identification of structural or behavioral differences.
- **Engagement Filtering** – A feature in NetXplore that highlights active participants while suppressing low-engagement noise to improve analytical clarity.

- **Report Generation** – The automated creation of a summary document that includes applied filters, visualizations, network metrics, and key findings.
- **Anonymization** – The systematic replacement or masking of participant identifiers to ensure privacy throughout analysis and reporting.

1.7.2. Generic Terminology

- **Node** – The smallest unit in a network graph, representing a participant in the conversation.
- **Edge** – A connecting link between two nodes that represents an interaction.
- **Directed Graph** – A graph where edges include directionality, indicating the initiator and recipient of an interaction.
- **Undirected Graph** – A graph where edges lack directionality, representing mutual connections.
- **Degree** – The number of edges connected to a node. In directed graphs, it is divided into in-degree (incoming) and out-degree (outgoing).
- **Centrality** – A set of measures used to identify the most important or influential nodes in the network.
- **Betweenness Centrality** – A measure of how often a node lies on the shortest paths between other nodes.
- **Closeness Centrality** – The reciprocal of the average shortest path distance from a node to all other nodes.
- **Eigenvector Centrality** – A measure of a node's influence that considers both the number and quality of its connections.
- **PageRank** – An algorithm that ranks nodes by simulating random navigation, assigning more importance to nodes connected by other highly ranked nodes.
- **Weighted Edge** – An edge that carries a numerical value representing the frequency, strength, or significance of an interaction.

2. Literature Survey

This chapter reviews relevant academic and practical literature that informs the development of NetXplore. It focuses on the challenges associated with analyzing large-scale conversational data across heterogeneous platforms, and on the range of computational and visualization solutions that have been proposed to address these challenges. The discussion highlights both the strengths and the limitations of existing approaches, laying the foundation for understanding how NetXplore advances the analysis and exploration of social discourse networks.

2.1 Problem Survey

The rapid growth of online communication platforms has resulted in unprecedented volumes of conversational data, which carry significant potential for understanding social dynamics, influence, and interaction patterns. However, the structural heterogeneity of these data presents fundamental challenges. Linear message logs, such as those exported from WhatsApp, present

conversations as a chronological sequence, whereas other platforms adopt formats that emphasize nested or threaded interactions. These representations, while intuitive for human readers, complicate systematic computational analysis and hinder cross-platform comparison.

Traditional methods such as manual reading, qualitative coding, and discourse analysis are limited in their scalability and often fail to capture structural properties such as network connectivity, community formation, or participant influence. As conversational datasets continue to grow in size and complexity, these limitations become increasingly pronounced. Researchers have therefore emphasized the need for computational frameworks capable of transforming raw conversational data into structured formats that enable quantitative exploration, visual interpretation, and comparative analysis across heterogeneous platforms (Chang, Chiam, Fu, Wang, Zhang, & Danescu-Niculescu-Mizil, 2020) [1].

2.2. Solution Survey

In response to these challenges, multiple methodological and technological approaches have been proposed. Natural language processing techniques have been applied to detect sentiment, classify topics, and identify conversational roles. Among these, topic modeling approaches such as Latent Dirichlet Allocation (LDA) have become particularly influential for uncovering hidden thematic structures in large text corpora (Blei, Ng, & Jordan, 2003) [2].

Network-based approaches, in contrast, represent interactions as graphs, enabling researchers to investigate properties such as centrality, clustering, and community detection. Several platforms have been developed to operationalize these approaches. General-purpose social network analysis tools such as Gephi provide advanced visualization and computation features, allowing researchers to manipulate networks and explore their structure in an interactive way (Bastian, Heymann, & Jacomy, 2009) [3].

NetXplore builds upon these contributions by providing an integrated platform that unifies data ingestion, normalization, graph-based analysis, and interactive visualization. It is designed to handle heterogeneous data sources, incorporate privacy-preserving mechanisms such as anonymization, and streamline the entire analytical workflow into a single environment. In doing so, it addresses the methodological and practical gaps identified in the literature and advances the capabilities available for network-oriented analysis of social discourse.

2.3 Interactive Graph Analysis

A central innovation of NetXplore lies in its transformation of automated data processing into an interactive user driven exploration environment. Conversational data are first preprocessed and then represented as dynamic network graphs, which researchers can explore to reveal structural patterns, anomalies, and relationships not visible in static representations.

The platform integrates automated computation of network metrics with real time interaction, enabling iterative hypothesis testing and flexible refinement of research focus. Instead of being constrained by fixed outputs, researchers can manipulate the graph according to their objectives. This is realized through capabilities such as dynamic filtering by date, participants, or keywords, highlighting of nodes and communities using centrality measures, side by side graph comparison, and customizable visual layouts. In addition, NetXplore supports the export of

tailored reports that combine analytical results with visual representations, preserving both methodological transparency and interpretive flexibility.

2.4 Platform Design and User Interaction

NetXplore is implemented as a web based research environment that integrates data ingestion, preprocessing, graph generation, and interactive exploration. Its design philosophy emphasizes accessibility, flexibility, and transparency, making advanced network analysis tools available to both novice and expert researchers.

The user interface, built with React, places the graph visualization at the center of the analytical workflow. All key tools including filtering, highlighting, and comparison are integrated into the same environment, minimizing the need for navigation between modules. The workflow is streamlined. Once data are uploaded, preprocessing and normalization are performed automatically, after which the interactive graph is immediately available for exploration.

Researchers can customize the visual presentation of graphs, apply structural metrics such as the Distance Weighted Ranking Algorithm, and compare multiple datasets to identify structural or behavioral differences. A dedicated research management component allows users to save, revisit, and compare past analyses, supporting iterative and longitudinal research.

By combining automated data handling with interactive visualization, NetXplore embodies a design philosophy that balances computational power with user agency, enabling researchers to derive meaningful insights while maintaining methodological rigor and flexibility.

3. Technological Survey

3.1. Problem Survey

The NetXplore project addresses the challenge of analyzing heterogeneous conversational data and transforming it into structured, network-oriented representations that can be explored interactively. This involves several technological challenges, including data parsing from diverse formats, scalable data storage and retrieval, efficient graph construction and analysis, real-time visualization, and collaborative research management. Specifically, the key challenges can be summarized as follows:

1. **Data Parsing and Normalization:** Conversational data is stored in different formats across platforms. WhatsApp exports are linear text logs, while Wikipedia talk pages adopt a hierarchical HTML structure. Parsing and normalizing such heterogeneous sources into a unified schema poses significant technical challenges.
2. **Graph Construction and Analysis:** Transforming conversations into social network graphs requires algorithms capable of identifying participants, reply relations, and

structural properties. In addition, computing meaningful network metrics and applying edge-weighting techniques is essential for revealing interaction patterns.

3. **Data Storage and Management:** Storing normalized conversation data alongside dynamically generated graph structures requires a robust and scalable database solution. Efficient retrieval and query management are necessary to support large-scale analysis without compromising performance.
4. **Visualization and Interactivity:** A key requirement of the project is to provide an intuitive and interactive interface that allows researchers to explore networks in real time, apply filters, compare datasets, and extract insights. This necessitates a visualization framework capable of rendering large graphs responsively in the browser.
5. **Collaboration and Version Control:** To support team-based research and iterative development, the project requires reliable version control and collaborative tools that enable transparent tracking of changes and streamlined deployment.

3.2. Solution Survey

To address these challenges, extensive research and evaluation of available technologies were conducted. The following components were selected for NetXplore:

1. Data Parsing and Normalization
 - BeautifulSoup was chosen for extracting structured content from Wikipedia talk pages in HTML format.
 - Regular Expressions (Regex) are employed to process WhatsApp TXT exports, allowing accurate identification of participants, timestamps, and reply relations.
 - Together, these tools ensure accurate ingestion and normalization of heterogeneous data into a unified schema suitable for graph construction.
2. Graph Construction and Analysis
 - NetworkX, a Python library for network analysis, was selected for constructing and analyzing conversational graphs.
 - It supports the computation of network metrics such as centrality, clustering, and community detection.
 - The Distance Weighted Ranking Algorithm was implemented to weight edges based on message sequence and proximity, enhancing the interpretability of reply dynamics.
3. Data Storage and Management
 - PostgreSQL serves as the primary relational database for storing normalized conversation data, graph structures, and user analyses.

- Database interactions are managed through Xata, which provides a flexible data management layer, simplifying queries and ensuring efficient retrieval of stored records.
4. Visualization and Interactivity
 - The frontend is developed with React, enabling a dynamic, modular, and user-friendly interface.
 - ForceGraph2D is integrated to provide real-time rendering of interactive network graphs directly in the browser.
 - Bootstrap is used for styling and responsive layouts, ensuring a clean and accessible user interface without requiring extensive custom CSS.
 5. Collaboration and Version Control
 - The project employs Git and GitHub for version control, supporting distributed collaboration, systematic change tracking, and integration with deployment pipelines.
 - These tools ensure transparency, reproducibility, and streamlined project management across the development lifecycle.

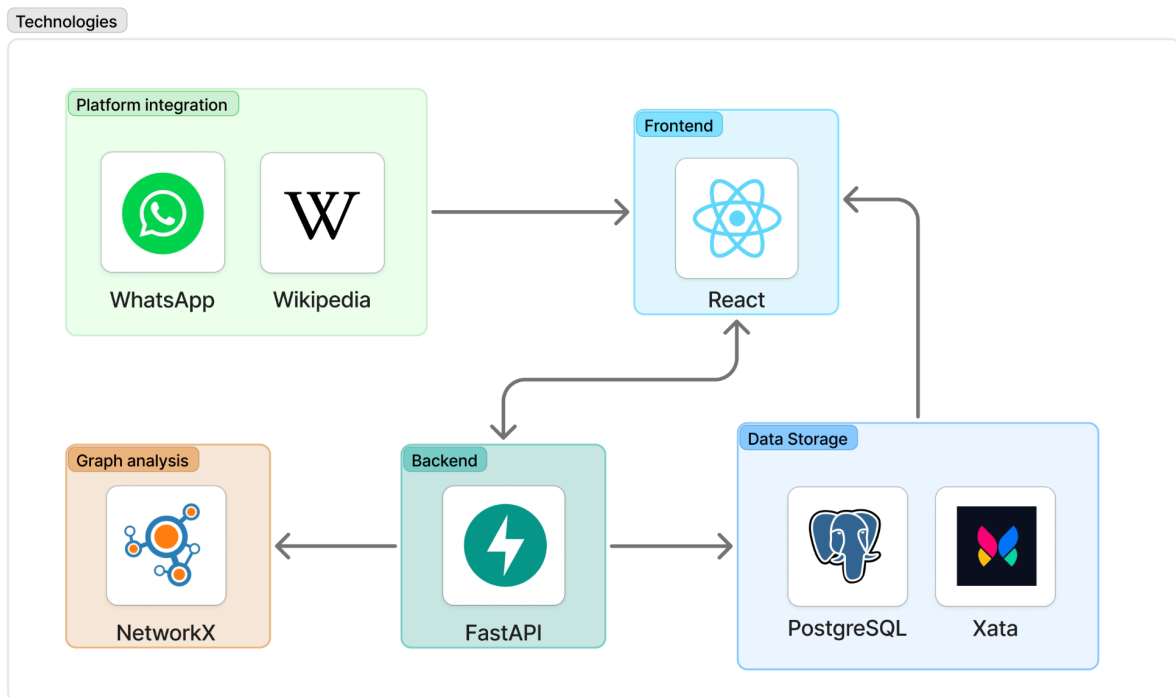


Figure 3.2 – Selected Technologies for NetXplore

3.3. Discussion and Conclusion

The technological choices for NetXplore were made with careful consideration of their ability to address the project's challenges. By combining BeautifulSoup and Regex for data parsing, NetworkX for graph analysis, PostgreSQL with Xata for data management, and React with

ForceGraph2D for visualization, the platform ensures an end-to-end workflow from data ingestion to interactive exploration.

These technologies together provide a scalable and extensible infrastructure that supports both methodological rigor and practical usability. The integration of automated preprocessing, graph-based analysis, and interactive visualization within a unified environment distinguishes NetXplore from existing solutions, advancing the state of the art in the study of conversational networks.

4. Requirements and Specification

This chapter defines the functionalities that NetXplore must deliver and the conditions it must satisfy to fulfil its role as a research platform for social network analysis. Functional requirements specify what the system shall do to be considered successful, whereas non-functional requirements detail the qualities of the implementation performance, security, maintainability and usability that ensure the functionality is delivered appropriately.

4.1 Functional Requirements

4.1.1 Researcher Platform

The NetXplore platform is built for researchers who wish to study communication networks derived from chat data. To support a variety of research designs and comparative analyses, the system incorporates several core capabilities:

- **User and Research Management:** Researchers must be able to register, authenticate and manage their accounts securely through password-based login and optional OAuth. They can create new research projects by specifying a title, description and chosen data source (WhatsApp export or Wikipedia talk page). The system stores project metadata creator, creation timestamp, selected platform and description to enable later retrieval and analysis.
- **Data Upload and Acquisition:** The platform shall support uploading exported WhatsApp chat logs in text format and Wikipedia discussion pages in JSON format, as well as retrieving a talk page directly given a URL. Upon upload, NetXplore validates the file structure and determines the correct parser to apply. If the file fails format checks, the system shall return an informative error rather than processing corrupted data. Researchers may choose whether to store the uploaded data persistently or process it only within their session.
- **Network Construction and Visualisation:** Following successful upload and parsing, NetXplore must construct a communication graph where nodes represent participants and edges represent consecutive messages or replies. Researchers can choose between directed and undirected graph models. Edge weights are derived from message sequences via a distance-weighted ranking algorithm, and the system computes node centrality measures (degree,

betweenness, closeness, eigenvector and PageRank) and graph-level measures such as density and reciprocity. An interactive 2D network visualisation in the browser shall display the resulting graph, with the ability to zoom, pan and inspect node metrics.

- **Filtering and Anonymisation:** The user interface must allow researchers to apply filters before graph construction: selecting date ranges, defining minimum or maximum message lengths, including or excluding specific keywords, and specifying a minimum number of consecutive messages required to form an edge. For privacy protection, the platform shall provide an anonymisation option that replaces participant names with consistent pseudonyms.
- **Community Detection:** NetXplore shall implement community detection algorithms Louvain, Girvan–Newman and greedy modularity maximisation to identify cohesive groups within the communication graph. For each detected community, the system computes modularity and summarises average centrality metrics of its members. The results are presented in tables and colour-coded within the network visualisation.
- **Network Comparison:** Researchers must be able to compare multiple saved network analyses, not merely two. The comparison shall compute and present differences in numbers of nodes and edges, relative densities, and overlaps among participant sets with percentage differences across all selected networks. A combined visualisation shall highlight shared nodes and edges across the networks using distinct colours, and a summary table shall aid interpretation.
- **Project History and Reporting:** Upon completing a network analysis, the platform does not persist the results automatically. To protect the privacy of personal conversations, researchers must explicitly choose whether to save the analysis under the current project. When saved, the system records the selected filters, node and edge lists and computed metrics and associates them with the research record. A dashboard lists all of a user's saved analyses with their creation dates, platforms and basic statistics. Within this dashboard researchers may rename, duplicate or delete analyses, and they can generate comprehensive reports including graphs, statistical tables and textual summaries for download as PDF or CSV files. Analyses that are not saved remain in session memory only and are removed when the session ends.

4.1.2 Data Processing Pipeline

- **Normalisation and Parsing:** The backend must include dedicated parsers for WhatsApp and Wikipedia data. These parsers extract message text, sender identifiers, timestamps and reply relationships, normalising date and time formats and removing spam. An optional anonymisation step assigns pseudonyms to participants. The pipeline ensures that only valid messages populate the graph and that metadata such as message length and temporal spacing are captured.

- **Platform-Specific Analysis:** A common abstract class defines the interface for analysis methods. Concrete implementations for WhatsApp and Wikipedia inherit from this base and employ appropriate parsing logic. For Wikipedia talk pages, the parser must translate the nested thread structure into a linear sequence of interactions while preserving reply hierarchies.
- **Metric Computation:** During graph construction, the system shall calculate edge weights based on sequential messages and adjust them according to temporal distance if the researcher opts to use the distance-weighted ranking. Node centrality and graph-level statistics are computed and returned to the frontend for display.
- **Data Storage:** Persisted analyses are stored in a PostgreSQL database through an asynchronous SQLAlchemy ORM. Tables include Research (project metadata), ResearchFilter (parameter settings), NetworkAnalysis (nodes, edges and metrics) and Comparison (results of network comparisons). Session-only analyses remain in memory and are discarded upon logout or session expiry.
- **File Handling:** The /upload endpoint verifies file extensions and inspects sample lines using regular expressions to detect the data source. A separate endpoint fetches and processes Wikipedia talk pages using HTTP requests and HTML parsing. Improperly formatted files are rejected with descriptive error messages.
- **Logging and Monitoring:** The backend shall record operational events, user actions and errors to facilitate auditing and debugging. Logs are restricted to authorised developers and do not expose sensitive data. Monitoring tools may be integrated to track performance metrics and resource utilisation.

4.3 Specification: The Scope of NetXplore

NetXplore is an advanced research platform designed to transform unstructured textual conversations into rich, analysable social networks. Its scope encompasses the entire workflow from importing raw chat exports through building directed or undirected graphs, computing sophisticated metrics (such as degree, betweenness, closeness, eigenvector centrality and PageRank), detecting communities and comparing multiple networks. By offering real-time filtering, anonymisation and interactive visualisations, the platform enables researchers to uncover communication patterns, identify influential participants and examine structural differences across experiments. NetXplore currently focuses on conversations from WhatsApp and Wikipedia talk pages but is architected to be extensible to other communication platforms. It intentionally separates network analysis from content-level sentiment or predictive modelling to ensure objectivity and reproducibility, while its modular design leaves room for future integration of such analyses. Through comprehensive dashboards and exportable reports, NetXplore provides a deep, user-controlled exploration of social networks that elevates research beyond simple metrics.

4.4 The NetXplore approach

The development of NetXplore follows a modular and platform-specific methodology that ensures accurate processing of heterogeneous data sources while maintaining flexibility and scalability. The system is composed of discrete but interconnected

components, each dedicated to a specific stage of the workflow. The workflow begins with Platform-Specific Parsing, which extracts data from WhatsApp and Wikipedia using dedicated parsers designed to respect the unique structures of linear and threaded conversations. Next, Data Preprocessing ensures methodological consistency and compliance with privacy standards through normalization and optional anonymization.

Once the data is prepared, Graph Construction and Analysis transforms it into interactive social network graphs, where researchers can select directed or undirected formats, apply weighting algorithms, and compute centrality metrics to gain both qualitative and quantitative insights. Building on this, Interactive Visualization enables dynamic exploration of the networks, allowing users to adjust filters, highlight communities, and examine participants in real time.

To support research continuity, Result Management provides flexibility by storing analyses in PostgreSQL for future retrieval and comparison, or by maintaining them within temporary sessions. Finally, Report Generation consolidates all findings into exportable documents, thereby supporting scholarly dissemination and academic publication. Through this modular approach, NetXplore offers a comprehensive yet accessible platform for conducting rigorous social network analysis on large-scale discourse data.

5. The Architecture

The architecture of NetXplore is designed to support the complete research workflow, from raw data ingestion to visualization, analysis, and reporting. It follows a modular and layered approach, separating backend data processing from frontend user interaction, and ensuring scalability, maintainability, and responsiveness.

To describe the architecture of the NetXplore system, we begin by outlining the heterogeneous data sources it processes, including WhatsApp conversations and Wikipedia talk pages, and the transformations applied to prepare them for analysis. We then define the roles supported within the system, such as researchers who conduct the analyses and administrators who manage platform operation. Finally, we present the sequence of activities performed during the research life cycle – from uploading or selecting data, through graph construction and interactive exploration, to saving and exporting results – providing a clear view of how NetXplore transforms raw discourse into structured analytical insights.

5.1 System Architecture

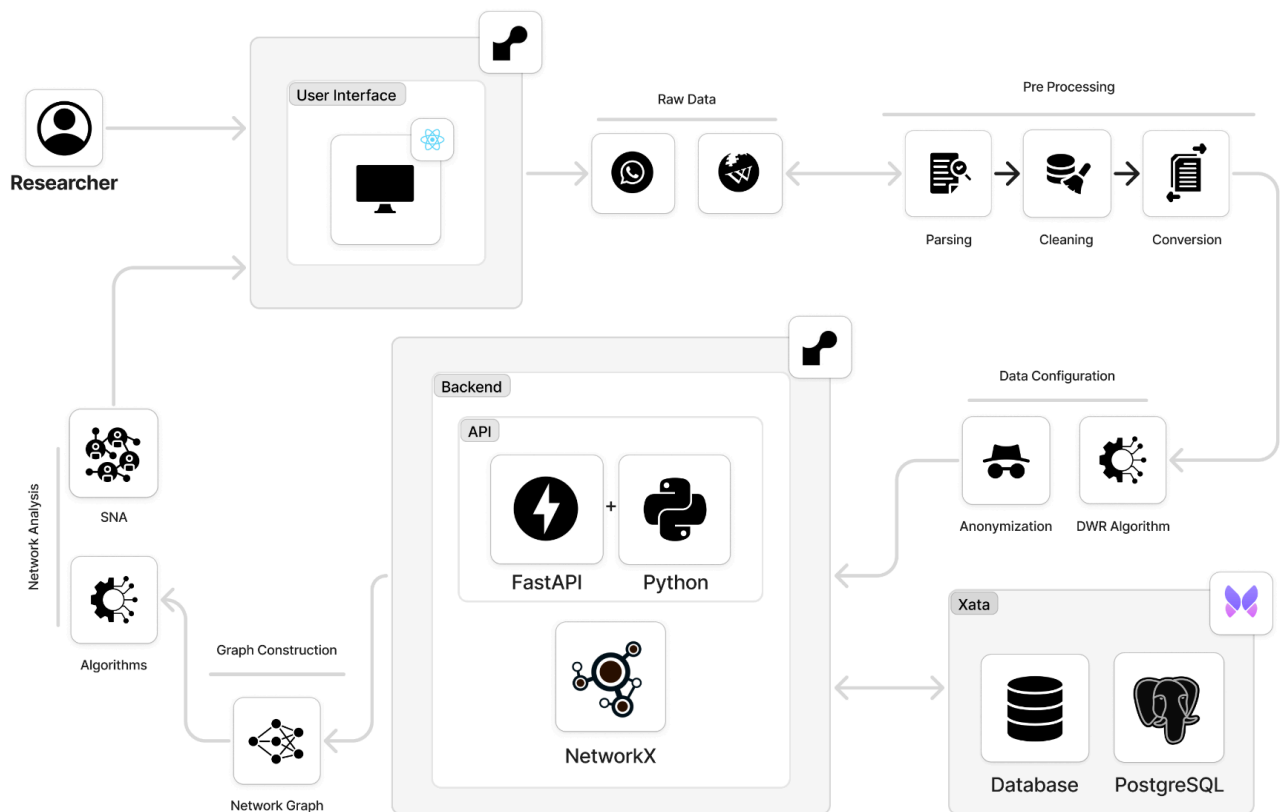


Figure 5.1 - System Architecture

5.2 NetXplore Data

Data Sources and Parsing: NetXplore supports two primary types of conversation datasets, each requiring a specialized parsing approach:

- WhatsApp Chat Exports:** These are plain text `.txt` files exported from WhatsApp chats, where each message line is prefixed by a timestamp and the sender's name, followed by the message content. The NetXplore parser for WhatsApp reads each line and extracts the date/time, participant name, and message text. It accounts for different timestamp formats (e.g. various date orders or the presence/absence of seconds) and ignores lines that don't represent actual messages such as system notifications or media placeholders (e.g. "<Media omitted>"). The result is a chronological list of user messages extracted from the chat log, ready for analysis.
- Wikipedia Talk Pages:** These are discussion pages on Wikipedia where contributors converse in threads. Users sign their comments with a username and a timestamp, and reply comments are typically indented with colons as a visual indicator. NetXplore uses a dedicated parser to handle this semi-structured format. The system fetches the talk page content (HTML) and identifies individual comments along with their author and timestamp. By analyzing the indentation levels and signature markers in the text, the

parser infers the reply structure essentially determining *who replies to whom*. This context-aware extraction preserves the threaded conversation structure (parent and reply comments) in the output dataset.

Preprocessing and Cleaning: After parsing the raw data from either source, NetXplore applies several preprocessing steps to clean and standardize the content before graph construction:

- **Cleaning:** The parser removes extraneous or irrelevant content that is not part of the actual dialogue. For WhatsApp chats, this includes filtering out system-generated lines (such as encryption notices or group system messages) and placeholders for media or deleted messages. For Wikipedia discussions, cleaning involves stripping wiki markup or template text from the comments and removing any administrative notes or artifacts (for example, edit signatures or repeated formatting) so that only the actual discussion text remains.

[illegible]

Example of filtering rules in NetXplore. The parser uses a regular expression (`MEDIA_RE`) to detect media placeholders and a curated list (`spam_messages`) to exclude system-generated or irrelevant line

- **Normalization:** NetXplore normalizes the data format to ensure consistency across all messages. This includes converting timestamps into a unified format (e.g. standardizing dates and times regardless of the original locale format) and normalizing user names or IDs. It also handles text encoding issues by removing non-printable characters or directional markers that might appear in exports. In the case of Wikipedia talk data, the threaded comments may be converted into a linear conversation log (with indicators of reply relationships) so that they can be processed in a similar way to chat logs. This normalization step guarantees that subsequent analysis treats all messages uniformly.

NetXplore

```
def detect_date_format(first_line: str) → list[str]:
    if re.search(r"[\d{1,2}[\.]\d{1,2}[\.]\d{4},\s\d{2}:\d{2}:\d{2}]", first_line):
        return ["%d.%m.%Y, %H:%M:%S", "%d.%m.%Y, %H:%M"]
    elif re.search(r"\d{1,2}/\d{1,2}/\d{2,4},\s\d{1,2}:\d{2}:", first_line):
        return ["%m/%d/%y, %H:%M", "%m/%d/%Y, %H:%M"]
    else:
        return ["%d/%m/%y, %H:%M", "%d.%m.%Y, %H:%M"]

def parse_date_time(date_str: str | None, time_str: str | None) → datetime | None:
    if not date_str:
        return None

    try:
        if time_str and time_str.strip() and time_str != "None":
            try:
                return datetime.strptime(f"{date_str} {time_str}", "%Y-%m-%d %H:%M:%S")
            except ValueError:
                return datetime.strptime(f"{date_str} {time_str}", "%Y-%m-%d %H:%M")
        else:
            return datetime.strptime(f"{date_str} 00:00:00", "%Y-%m-%d %H:%M:%S")
    except ValueError as e:
        print(f"Error parsing date/time: '{date_str}' + '{time_str}' - {e}")
        raise ValueError(f"Invalid date/time format. Expected format: YYYY-MM-DD and HH:MM:SS. Got: '{date_str}' + '{time_str}'")
```

Example implementation of NetXplore's timestamp normalization pipeline. The `detect_date_format` function identifies locale-specific timestamp patterns, while `parse_date_time` parses and converts them into a consistent standardized format.

- **Anonymization (Optional):** Depending on researcher preferences, NetXplore can anonymize participant identities in the dataset. When this option is enabled, the actual usernames are replaced with generic labels (e.g. *User_1*, *User_2*, etc.) throughout the data. This step retains the structure of the conversations (you can still distinguish who is who by consistent labels) but removes personal identifiers, which is useful for privacy or unbiased analysis.

```
if anonymize:
    if user not in anonymized_map:
        anonymized_map[user] = f"User_{len(anonymized_map) + 1}"
    user = anonymized_map[user]
    if reply_to and reply_to not in anonymized_map:
        anonymized_map[reply_to] = f"User_{len(anonymized_map) + 1}"
    reply_to = anonymized_map.get(reply_to, reply_to)
```

Implementation of anonymization in NetXplore. Real participant names are replaced with consistent pseudonyms (e.g., *User_1*, *User_2*), applied both to message senders and reply references.

Prepared Data for Graphs: The output of the above parsing and preprocessing pipeline is a clean, structured set of conversation data ready for network analysis. Each entry in this processed dataset represents a message with a clear timestamp, a normalized (or anonymized) user identifier, and the message text, along with context links such as reply targets (in the case of threaded discussions). This processed data forms the foundation for constructing interactive social network graphs. In these graphs, each participant becomes a node, and each interaction (message exchange or reply) becomes an edge between nodes. The edges can be weighted by the frequency or sequential flow of messages, reflecting the intensity of interaction between each pair of participants. In short, by meticulously parsing and refining the raw conversation logs, NetXplore ensures that the subsequent network graph accurately mirrors the conversation's structure and content, providing a reliable basis for further social network analysis.

```
G = nx.DiGraph() if directed else nx.Graph()
G.add_nodes_from(usernames)

for edge, weight in edges_counter.items():
    if edge[0] in usernames and edge[1] in usernames:
        G.add_edge(edge[0], edge[1], weight=weight)

if directed:
    is_connected = nx.is_weakly_connected(G) if len(G.nodes()) > 0 else False
else:
    is_connected = nx.is_connected(G) if len(G.nodes()) > 0 else False

try:
    degree centrality = nx.degree_centrality(G)
    betweenness centrality = nx.betweenness_centrality(G, weight="weight")

    if is_connected and len(G.nodes()) > 1:
        closeness centrality = nx.closeness_centrality(G)
        eigenvector centrality = nx.eigenvector_centrality(G, max_iter=1000)
        pagerank = nx.pagerank(G)
    else:
        if len(G.nodes()) > 1:
            if directed:
                components = list(nx.weakly_connected_components(G))
            else:
                components = list(nx.connected_components(G))

            if components:
                largest_cc = max(components, key=len)
                subgraph = G.subgraph(largest_cc).copy()

                closeness centrality = nx.closeness_centrality(subgraph)
                eigenvector centrality = nx.eigenvector_centrality(subgraph, max_iter=1000)
                pagerank = nx.pagerank(subgraph)

                for node in G.nodes():
                    if node not in largest_cc:
                        closeness centrality[node] = 0.0
                        eigenvector centrality[node] = 0.0
                        pagerank[node] = 0.0
            else:
                closeness centrality = {node: 0.0 for node in G.nodes()}
                eigenvector centrality = {node: 0.0 for node in G.nodes()}
                pagerank = {node: 1.0/len(G.nodes()) for node in G.nodes()}
        else:
            closeness centrality = {}
            eigenvector centrality = {}
            pagerank = {}

except Exception as e:
    logger.error(f"Error calculating centrality measures: {e}")
    degree centrality = {}
    betweenness centrality = {}
    closeness centrality = {}
    eigenvector centrality = {}
    pagerank = {}

nodes_list = [
    {
        "id": user,
        "name": user,
        "group": 1,
        "messages": user_message_count.get(user, 0),
        "degree": round(degree centrality.get(user, 0), 4),
        "betweenness": round(betweenness centrality.get(user, 0), 4),
        "closeness": round(closeness centrality.get(user, 0), 4),
        "eigenvector": round(eigenvector centrality.get(user, 0), 4),
        "pagerank": round(pagerank.get(user, 0), 4)
    }
    for user in usernames
]
```

Example implementation of graph preparation in NetXplore. Each participant is added as a node, interactions are represented as weighted edges, and core network metrics (degree, betweenness, closeness, eigenvector centrality, PageRank) are computed to support subsequent social network analysis.

5.2.1 Researcher Data

In addition to conversational data, NetXplore maintains structured datasets describing researchers, their projects, and the results of their analyses. These data are stored in relational form using PostgreSQL and synchronized with Xata for cloud-based scalability and accessibility. This dual storage design ensures both consistency in structured queries and flexibility in distributed usage scenarios.

Each researcher is represented in the users table, which records profile information such as name, email address, and securely hashed authentication credentials. This foundation supports

account creation, secure login, and persistent association of research activities with their respective owners.

```
class User(Base):
    __tablename__ = "users"

    user_id = Column(UUID(as_uuid=True),
primary_key=True, default=uuid.uuid4)
    name = Column(String, nullable=False)
    email = Column(String, unique=True,
nullable=False)
    password = Column(String, nullable=True)
    avatar = Column(String, nullable=True)
    created_at = Column(DateTime(timezone=True),
default=lambda: datetime.now(pytz.utc))
```

The users table is implemented using SQLAlchemy as shown above. Each researcher is represented by a unique user_id (UUID), along with attributes such as name, email, password, avatar, and created_at. Constraints ensure that the email field is unique and required, while secure password handling supports authentication. This table provides the foundation for managing researcher accounts, enabling secure login, persistent identity, and linking users to their research projects within the system.

Project-level information is organized within the research table, where each entry contains metadata including the project title, description, creation date, and the originating platform (WhatsApp or Wikipedia). The messages table stores parsed conversational data, with each row representing an individual message. Attributes include the sender identifier, timestamp, content of the message, and when applicable reply relations that reconstruct the conversational hierarchy.

```
class Research(Base):
    __tablename__ = "research"

    __table_args__ = (
        CheckConstraint(
            "platform IN ('whatsapp', 'wikipedia')",
            name="valid_platform_check"
        ),
    )

    research_id = Column(UUID(as_uuid=True), primary_key=True, default=uuid.uuid4)
    research_name = Column(String, nullable=False)
    description = Column(String)
    platform = Column(
        String,
        nullable=False,
        server_default='whatsapp'
    )
    user_id = Column(UUID(as_uuid=True), ForeignKey("users.user_id"), nullable=True)
    created_at = Column(DateTime(timezone=True), default=lambda: datetime.now(pytz.utc))
```

Each project is stored with a unique research_id, a title, description, creation date, and its source platform (WhatsApp/Wikipedia). The table also links to the owning researcher (user_id) and enforces valid platform values through a check constraint. This structure ensures that all projects are consistently documented and easily retrievable for further analysis.

The outcomes of analytical processes are preserved in the network_analysis table. This table captures both graph level metrics, such as density and diameter, and node-level measures, such as degree or betweenness centrality. It may also store serialized graph representations to enable subsequent retrieval and visualization. To guarantee reproducibility, filter configurations applied during analyses such as time intervals, user selections, or keyword-based subsets are recorded in the research_filters table. Finally, comparative analyses are documented in the comparisons table, which links multiple datasets and records the parameters of longitudinal or cross-platform

studies.

```
class NetworkAnalysis(Base):
    __tablename__ = "network_analysis"

    id = Column(UUID(as_uuid=True), primary_key=True, default=uuid.uuid4)
    research_id = Column(UUID(as_uuid=True), ForeignKey("research.research_id"), nullable=True)
    nodes = Column(JSONB, nullable=False)
    links = Column(JSONB, nullable=False)
    metric_name = Column(String, nullable=True)
    is_connected = Column(Boolean, default=True)
    created_at = Column(DateTime(timezone=True), default=lambda: datetime.now(pytz.utc))
    communities = Column(JSONB, nullable=True)
```

Implementation of the `network_analysis` table in NetXplore. It stores graph-level and node-level metrics, serialized graph structures, and communities detected for subsequent retrieval and visualization.

Together, these interrelated datasets establish a comprehensive data backbone that links researchers to their projects, stores both raw messages and derived analyses, and ensures that results remain reproducible and transparent.

5.2.2 Initial Start-up Data

When initiating a new study, NetXplore requires the researcher to provide an input dataset and minimal metadata that anchors the project within the system. For WhatsApp, the researcher uploads a TXT export of the chat, whereas for Wikipedia the input is provided as a direct URL link to the relevant Talk Page section. In both cases, the researcher also supplies basic project metadata, including a title and description, which ensures that the project can be uniquely identified within the system.

Once the dataset or URL is submitted, NetXplore automatically creates a new entry in the research table and links it to the authenticated researcher in the users table. From this stage onward, the project becomes part of the research lifecycle: it appears in the history view, where the researcher can revisit past analyses, refine or edit project metadata, and review the results of comparative analyses that were previously conducted. The history functionality also allows for deletion of projects that are no longer relevant. Importantly, while comparisons can be reviewed in the history view, new comparisons must be initiated as part of the active research workflow rather than from the history interface.

This start-up phase therefore not only initializes each project but also defines the framework for its lifecycle within NetXplore. By combining user-supplied input (TXT or URL) with descriptive metadata, the system ensures that research projects are properly documented, reproducible, and manageable from their inception through their eventual review or archival.

```

from .whatsapp_analyzer import WhatsAppAnalyzer
from .wikipedia_analyzer import WikipediaAnalyzer
from .base_analyzer import BaseAnalyzer

def get_analyzer(platform: str) → BaseAnalyzer:
    platform = platform.lower()
    if platform == "whatsapp":
        return WhatsAppAnalyzer()
    elif platform == "wikipedia":
        return WikipediaAnalyzer()
    else:
        raise ValueError(f"Unsupported platform: {platform}")

```

Implementation of the Factory Pattern in NetXplore. Based on the project's input platform (WhatsApp or Wikipedia), the system dynamically returns the appropriate analyzer, ensuring that each dataset is processed with a parser tailored to its structure.

5.2.3 Data Collection

After the initial start-up phase, NetXplore proceeds to the systematic collection and structuring of conversational data. This stage transforms the raw input either a WhatsApp TXT export or a Wikipedia Talk Page URL into a set of structured records that can be persistently stored and later analyzed.

For WhatsApp datasets, the system parses the chat log line by line. Each message entry is decomposed into its core components: timestamp, sender identifier, and message content. The parser accounts for variations in date and time formats and filters out non-conversational entries, such as system notifications or media placeholders. Each cleaned message is then inserted into the messages table, where it is chronologically ordered to preserve the natural flow of the dialogue.

Wikipedia datasets follow a different pathway. When a Talk Page URL is provided, the system retrieves the discussion content and segments it into discrete messages. Contributors are identified through their signatures, and timestamps are extracted from the page text. Indentation levels and reply markers are analyzed to infer parent child relationships between comments. These relations are explicitly stored in the messages table as reply references, thereby preserving the hierarchical structure of the discussion while enabling it to be represented in a graph compatible format.

Across both platforms, each stored message entry includes a unique sender reference, a standardized timestamp, the textual content, and, when applicable, a link to the message it replies to. This structured representation ensures that conversational context is not lost during preprocessing. At the same time, it provides a consistent schema across heterogeneous data sources, allowing WhatsApp and Wikipedia datasets to be analyzed under the same methodological framework.

The Data Collection stage therefore bridges the gap between raw conversational logs and analytical readiness. By transforming unstructured inputs into structured database records, NetXplore guarantees that all subsequent stages-graph construction, metric computation, and visualization are grounded in reliable and reproducible data.

5.3 NetXplore Process

When a researcher uploads a dataset either a WhatsApp chat transcript (plaintext `.txt`) or a Wikipedia Talk page, the NetXplore system first identifies the source type. It then routes the data through a factory-pattern dispatcher that selects the appropriate analyzer module for parsing.

The platform-specific analyzer (either `WhatsAppAnalyzer` or `WikipediaAnalyzer`) implements a common interface to ensure consistent output structure. At this stage, the analyzer ingests and cleans the raw message data, extracting structured records of each message (timestamp, sender, content) while filtering out irrelevant entries such as media placeholders or system messages.

Once cleaned, these structured messages are stored in the database or in-memory structure, preserving an immutable record of the conversation. This design ensures reproducibility, as researchers can re-run analyses or audit the same data at any point.

After storage, the analyzer invokes the `GraphBuilder` to translate message exchanges into a social network graph. Each participant becomes a node, and interactions (replies, quotes, or sequential exchanges) form the edges that connect them.

Finally, NetXplore computes a suite of network metrics using `NetworkX`. Nodes are annotated with individual properties (e.g., degree, betweenness, closeness, PageRank), and the graph's overall structure is analyzed. The processed graph is then sent to the frontend for visualization and interactive exploration.

This modular pipeline from upload and parsing to graph construction and metric calculation emphasizes both modularity (separate parsers with shared graph logic) and reproducibility (consistent outputs and revisitable analyses).

5.3.1 Graph Construction and Metrics

The graph construction process in NetXplore is handled by the GraphBuilder component. This module assembles a NetworkX graph (directed or undirected, depending on the configuration) from the parsed conversation data. For directed settings, the graph captures the direction of replies or references, while undirected mode treats all interactions as mutual links. Each unique participant, whether a WhatsApp user or a Wikipedia contributor, is represented as a node, and every detected interaction produces an edge between the relevant nodes.

Edges are added both for explicit relationships (such as replies or quotes) and for implicit sequential interactions, where one message directly follows another user's message. To reflect conversational dynamics, edges can be weighted using the `calculate_sequential_weights` utility. This sliding-window approach assigns higher weights to responses that occur within a short message distance, allowing researchers to tune responsiveness and interaction intensity by adjusting the window size (n).

After the graph topology (nodes and weighted edges) is built, the system computes a rich set of network metrics using NetworkX. These include classical centrality measures:

- Degree centrality – number of direct connections each participant has.
- Closeness centrality – how close a participant is to all others, reflecting their accessibility within the network.
- Betweenness centrality – identifying brokers that sit on communication paths between others.
- Eigenvector centrality and PageRank – highlighting participants with influence not only through their connections, but also through how well-connected their neighbors are.

Each node record is annotated with these values, along with message counts and other metadata. This provides a quantitative profile of every participant in the conversation.

Finally, these metrics allow researchers to explore and compare interaction networks. For example, they can identify highly influential users in a WhatsApp group, or contrast network structures between WhatsApp and Wikipedia datasets. They also enable longitudinal studies, tracking how centrality scores evolve over time. By applying the same set of measures consistently across datasets, NetXplore supports systematic cross-platform and temporal analyses of social interaction.

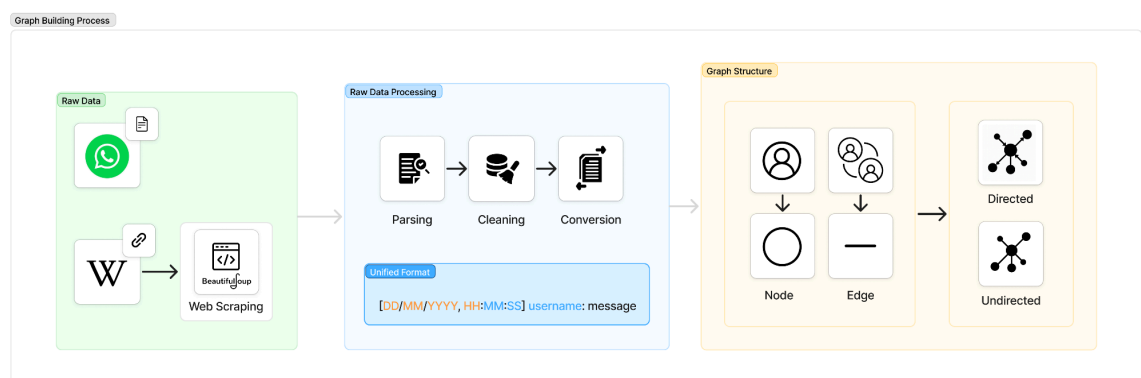


Figure 5.3.1 – Graph Building Process of NetXplore

5.3.2 NetXplore Network Analysis Process

Once the conversational graph has been constructed, NetXplore advances to the analytical phase. This stage leverages the NetworkX library together with custom utility functions to compute a range of structural, community, and global metrics, transforming the raw graph into a meaningful map of interaction patterns.

Node-Level Metrics

Each participant (node) in the network is enriched with multiple centrality scores that describe their relative role:

- **Degree Centrality** – quantifies the number of direct ties each participant maintains, highlighting active communicators.
- **Closeness Centrality** – measures how quickly a participant can reach all others, reflecting accessibility and reach.
- **Betweenness Centrality** – identifies “bridge” users who lie on many shortest paths, serving as brokers or gatekeepers of information.
- **Eigenvector Centrality & PageRank** – extend beyond direct ties by factoring in the importance of neighbors, surfacing influential participants who are embedded in well-connected neighborhoods.

These node-level annotations allow researchers to identify leaders, brokers, and peripheral users, enabling targeted interpretation of community roles.

Community Detection

Beyond individuals, NetXplore applies community detection algorithms (such as Louvain or Girvan–Newman, depending on configuration). These techniques partition the network into clusters of nodes that interact more intensively with one another than with outsiders. The detection of such subgroups is crucial for revealing conversational coalitions — for example, factions in a Wikipedia debate or cliques in a WhatsApp group.

Communities are stored directly in the database alongside the network analysis record, ensuring reproducibility and enabling side-by-side comparison across projects or over time.

Global Graph Metrics

At the overall structural level, NetXplore evaluates properties that describe the shape and cohesion of the conversation network:

- **Density** – the proportion of realized connections relative to all possible ones, reflecting overall interconnectedness.
- **Reciprocity** – in directed networks, measures the symmetry of exchanges (i.e., how often replies are mutual).
- **Diameter and Average Path Length** – characterize the “distance” across the network, indicating how quickly information might flow.
- **Modularity** – quantifies the strength of community division, useful for comparing how fragmented or cohesive discussions are across datasets.

Output and Reuse

All computed metrics from individual centralities to community labels and global measures are persisted in the `network_analysis` table. This guarantees that results are reproducible and that historical analyses can be reloaded without recalculation. Furthermore, storing both node- and

graph-level properties allows longitudinal and cross-platform comparisons: a researcher can, for example, track how betweenness centrality of a moderator evolves across months, or contrast the density of informal WhatsApp chats with formal Wikipedia debates.

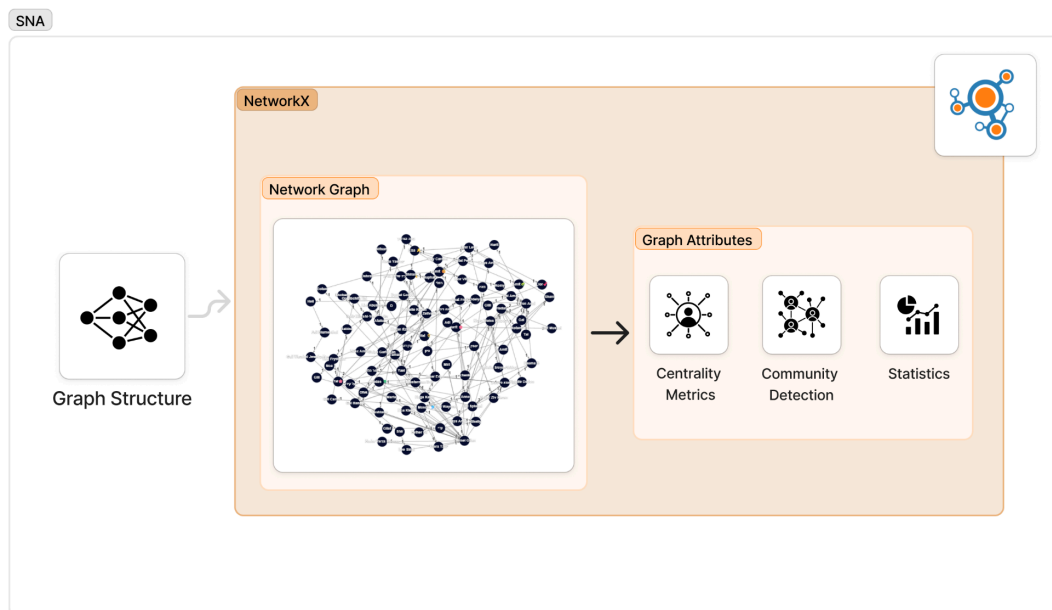


Figure 5.3.2 – Social network analysis process of NetXplore

5.3.3 Filtering, Highlighting, and Comparison Process

NetXplore incorporates a set of advanced mechanisms for refining and analyzing conversational networks, enabling researchers to adjust the scope of visualization and focus on specific interaction patterns. These mechanisms are divided into three core functionalities: filtering, highlighting, and comparison.

Filtering

The filtering capabilities allow researchers to narrow down the dataset according to multiple criteria. Typical filters include temporal constraints such as start and end dates, which make it possible to analyze conversations within a defined time window. Content-based filters are also supported, such as keyword matching that restricts the dataset to messages containing particular terms. Additional options include message length thresholds, filtering by response time, and anonymization of user identifiers to preserve privacy while maintaining network consistency. At the structural level, participants (nodes) can be temporarily removed from the visualization in order to evaluate how the exclusion of specific actors affects the overall network topology.

Highlighting

In addition to filtering, the system supports dynamic highlighting of nodes and edges. Nodes can be highlighted according to centrality measures such as degree, betweenness, or closeness, which enables the rapid identification of influential users or structural brokers. Alternatively, communities identified by algorithms such as Louvain or Girvan–Newman can be emphasized,

making it easier to distinguish clusters of participants within the graph. Highlighting thus adds an interpretive layer to the visualization, assisting researchers in identifying roles, subgroups, or bridges within the conversational network.

Comparison

NetXplore also facilitates the comparison of multiple networks side by side. This feature is particularly useful for examining temporal evolution, such as comparing the structure of the same group's interactions across different time periods, or for cross-platform studies that contrast WhatsApp conversations with Wikipedia debates. Comparative analysis is supported both visually, through simultaneous graph displays, and analytically, through comparative metrics such as density, modularity, and centrality distributions. All results of filtering and comparison can be stored in the system's database, ensuring reproducibility, and exported as structured research reports for further dissemination.

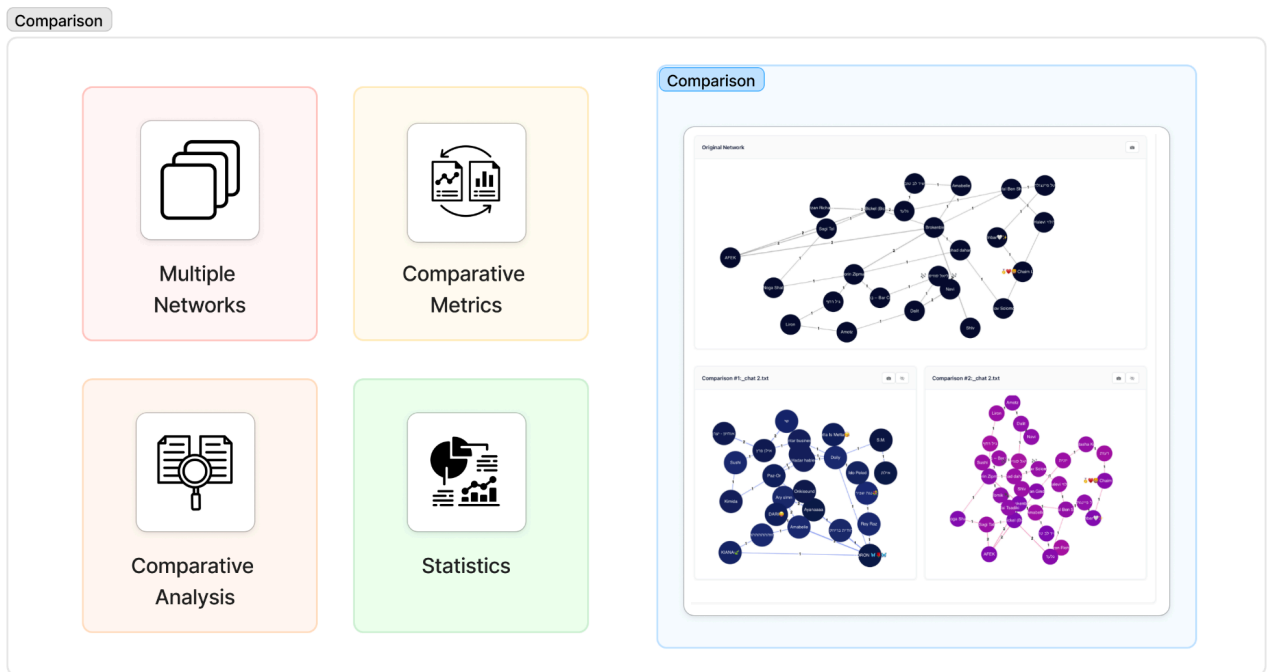


Figure 5.2.3 – Comparison Process of NetXplore

5.4 The life-cycle of a system

NetXplore operates as a modular research-driven platform for analyzing, comparing, and visualizing social discourse across heterogeneous data sources such as WhatsApp conversations and Wikipedia talk pages. It is designed to support researchers in transforming raw conversations into structured communication networks, uncovering interaction patterns, and generating meaningful insights that can be preserved, compared, and disseminated. The system lifecycle is composed of six distinct stages, each corresponding to a key component of the analytical workflow.

1. Data Ingestion

The lifecycle begins when the researcher uploads a WhatsApp export file (.txt) or selects a discussion section from a Wikipedia talk page. This step ensures heterogeneous sources are captured in a unified framework ready for processing.

2. Preprocessing

In this stage, the system normalizes the input data and, if chosen, anonymizes participant identifiers to ensure privacy. Filters such as date ranges, keywords, or participant thresholds can also be applied to tailor the dataset to the researcher's needs.

3. Graph Construction

The preprocessed data is transformed into a social network graph. Participants are represented as nodes and interactions as edges. The system supports directed or undirected graphs and applies weighting schemes that reflect communication intensity.

4. Analysis

NetXplore computes a range of network metrics, including degree, betweenness, and closeness centrality, and applies community detection algorithms such as Louvain to identify clusters of participants. This provides both quantitative measures of influence and qualitative insights into group structures.

5. Visualization & Exploration

The constructed network is rendered in an interactive interface. Researchers can zoom, filter, and highlight communities or individuals, exploring the conversation dynamically. The visualization reveals patterns and structures not visible through manual reading of the raw data.

6. Report Generation

Finally, results are consolidated into exportable reports. Researchers can save their studies in the system's database, revisit past analyses, compare multiple datasets, or export findings to CSV for scholarly dissemination and presentation.

Through this lifecycle, NetXplore provides a complete end-to-end framework for transforming raw, unstructured conversations into rigorous social network analyses, bridging qualitative exploration with quantitative measurement.

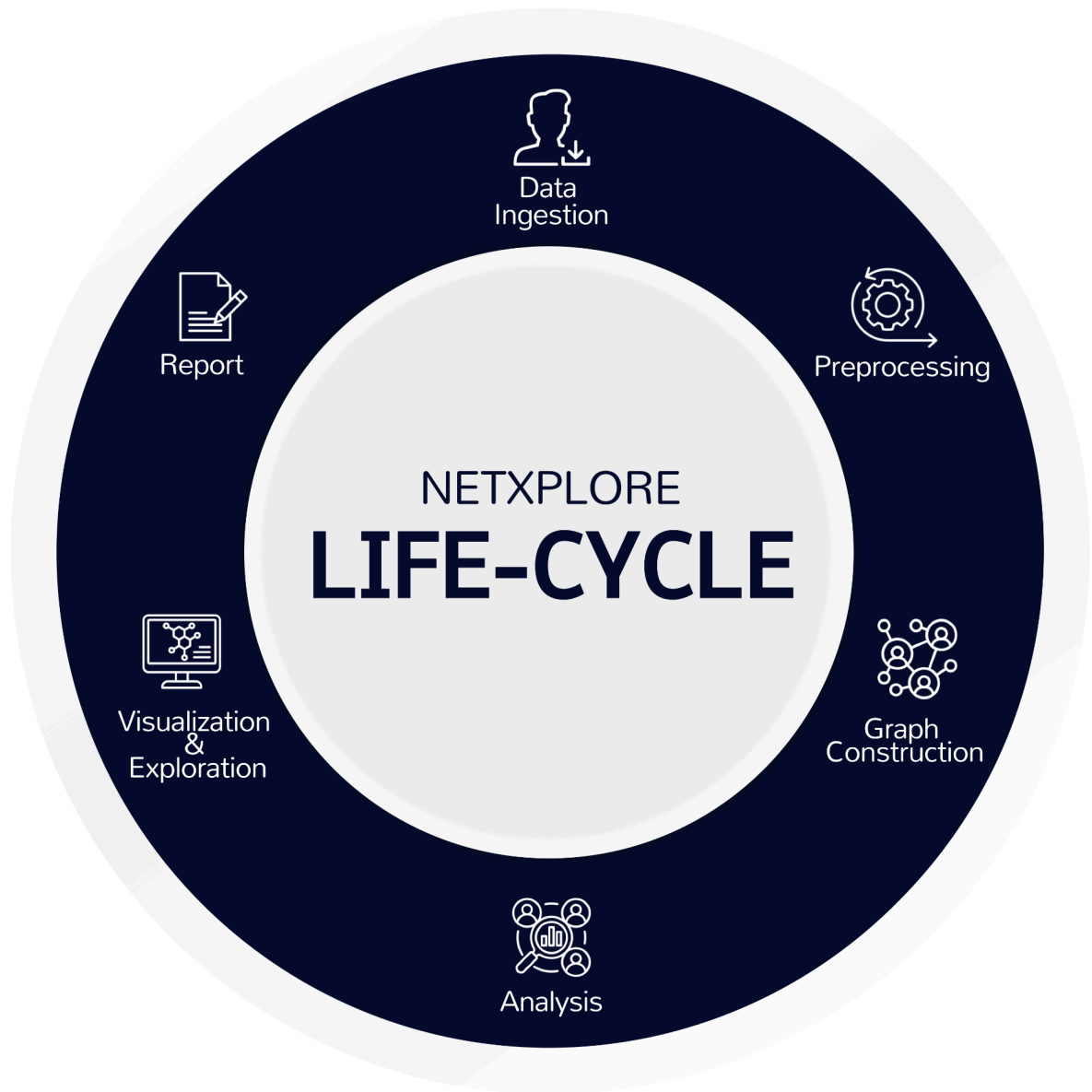


Figure 5.6 – System life-cycle of NetXplore

5.5. Roles

The NetXplore platform is designed primarily for researchers and analysts who study social interactions and discourse across digital platforms such as WhatsApp and Wikipedia. These users are responsible for uploading conversation datasets (e.g., exported chat logs or structured discussion pages) and initiating the processing pipeline.

Once data is uploaded, researchers can trigger multiple workflows: graph construction, community detection, centrality analysis, and comparative studies across platforms or timeframes. The system does not function independently but responds to the researcher's inputs such as applying filters, defining time ranges, or selecting subsets of users to focus on.

Through the interactive graph interface, researchers can visualize social structures, detect clusters of activity, and explore metrics such as degree, betweenness, and eigenvector centrality. They may also perform comparative analysis between different datasets, allowing them to identify behavioral shifts, communication patterns, or anomalies in discourse.

Finally, researchers can store their studies in the PostgreSQL database (via Xata integration), enabling them to revisit projects, refine their analysis, and generate exportable reports for academic or organizational use. This role-centric design ensures that NetXplore empowers researchers to uncover insights about social networks in a structured, reproducible, and scalable manner.

6. SYSTEM DESIGN

The system design of NetXplore defines the detailed structure, functionality, and operational workflow of the platform. It translates the architectural vision into concrete design specifications that govern how the system processes data, manages resources, and delivers results to researchers. The design covers component responsibilities, data flow, database schema, API definitions, security mechanisms, and system management features. The emphasis is on modularity, extensibility, and reliability, ensuring that NetXplore can accommodate current requirements while remaining adaptable to future research needs.

6.1 Use Cases

The use case diagram of NetXplore illustrates the core interactions of the Researcher with the system. The Researcher creates studies, uploads datasets, applies filters, and triggers analyses that generate interaction graphs and metrics. They can explore and customize visualizations, compare results, generate reports, and manage their research history, supporting both exploratory and reproducible workflows.

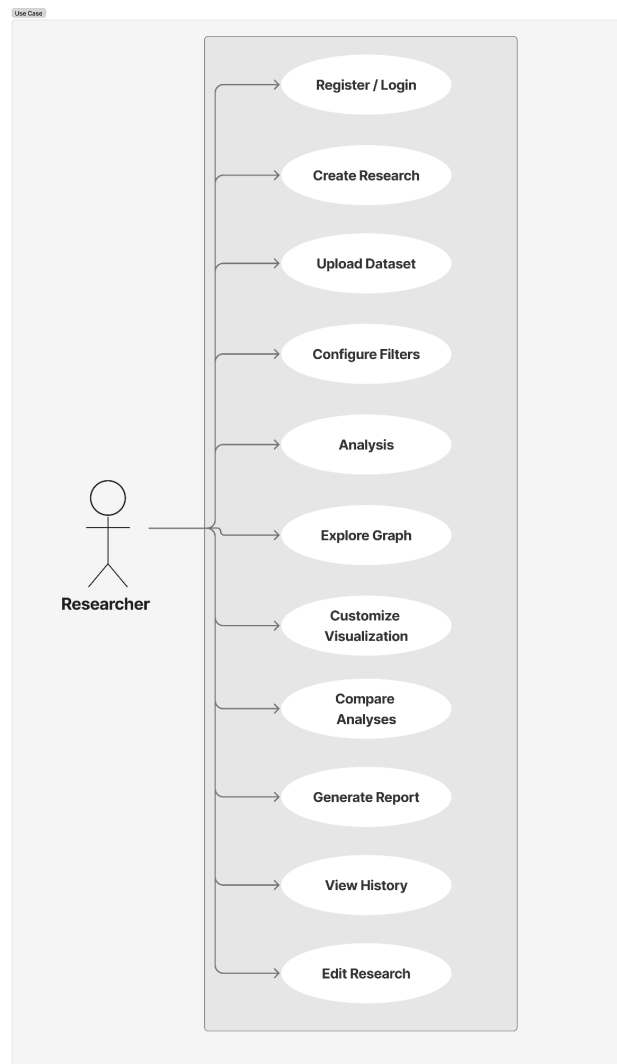


Figure 6.3.1 – Use-Case diagram of NetXplore

6.2 Class Diagram

The structural design of NetXplore is represented by a class diagram that illustrates the main system entities and the relationships between them. The diagram highlights how users are connected to research projects, how each research includes messages and filters, and how analyses and comparisons are derived from these projects. This view provides a high-level understanding of the system's architecture and the logical connections between its core components.

Class Diagram

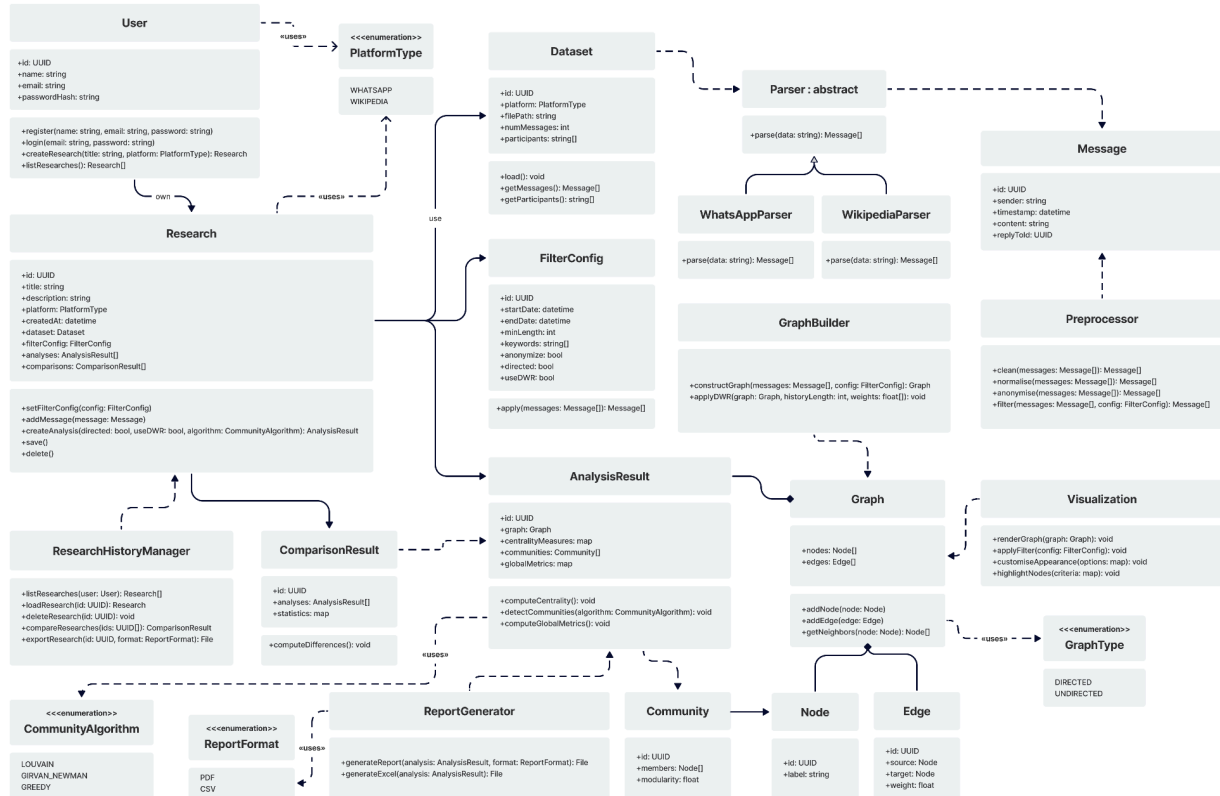


Figure 6.2 – Class diagram of NetXplore

6.3 Data Components

The NetXplore database is structured around several interconnected entities that together enable storing, processing, and visualizing conversational data. The users entity manages researcher accounts and authentication, while the research entity represents individual projects that connect datasets, filters, and analytical results. Raw conversational content from WhatsApp or Wikipedia is stored in messages, and configurable parameters such as anonymization, keywords, and time ranges are recorded in research filters. The outcomes of social network analysis, including graphs, metrics, and detected communities, are captured in network analysis, and cross-project or longitudinal evaluations are maintained in comparisons. Collectively, these components form the backbone of NetXplore's data model, ensuring scalability, reproducibility, and structured management of research workflows.

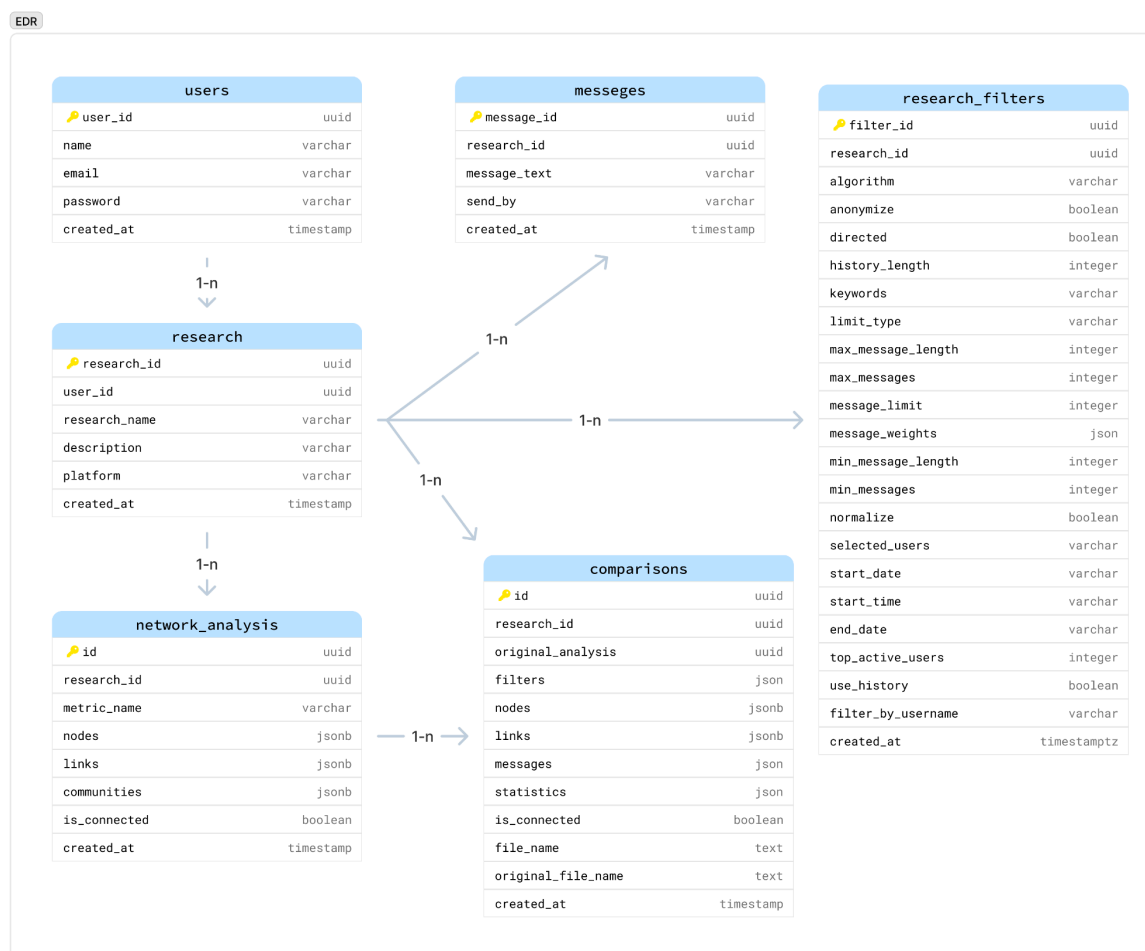


Figure 6.3 – Data Components EDR diagram of NetXplore

6.4 Interactions

This section presents different diagrams to describe the interaction design of NetXplore.

6.4.1 Activity diagram

The activity architecture of NetXplore describes the main workflow that guides the researcher from data ingestion to results visualization.

[Activity diagram](#)

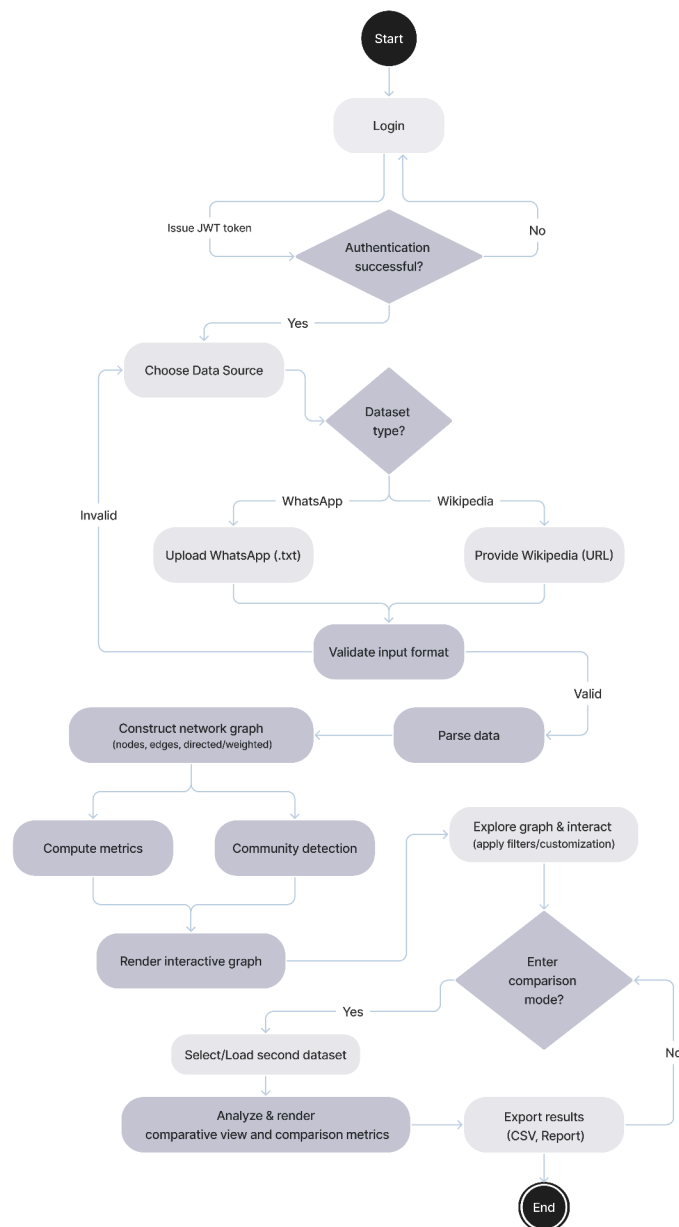


Figure 6.4.1 - Activity Architecture of NetXplore

6.4.2 Sequence Diagram

The following sequence illustrates the main interaction flow in NetXplore:

[Sequence Diagram Data Upload to Graph Build](#)

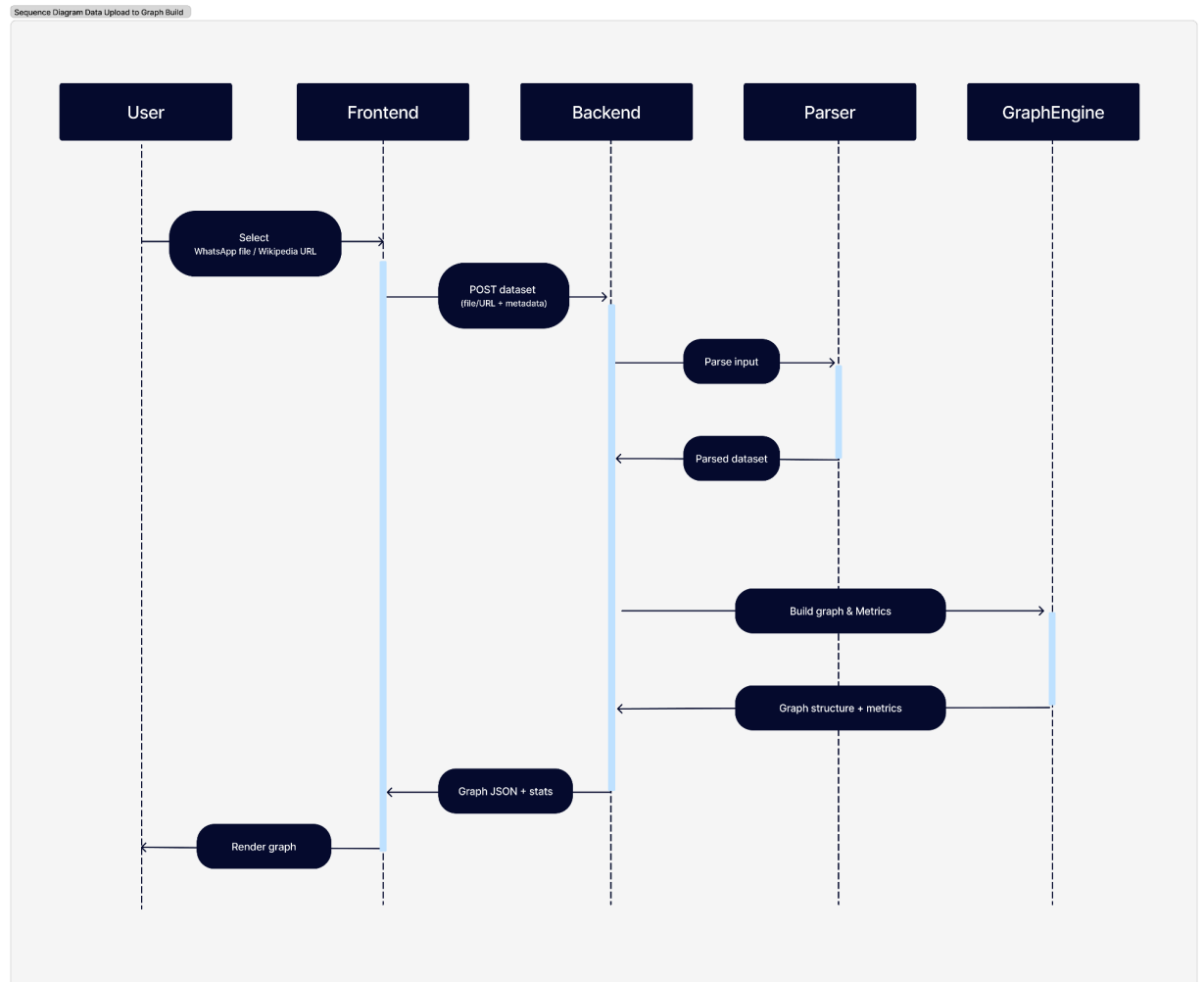


Figure 6.4.2-1 – Sequence diagram of NetXplore data upload and graph construction

Sequence Diagram Filter / Update Parameters

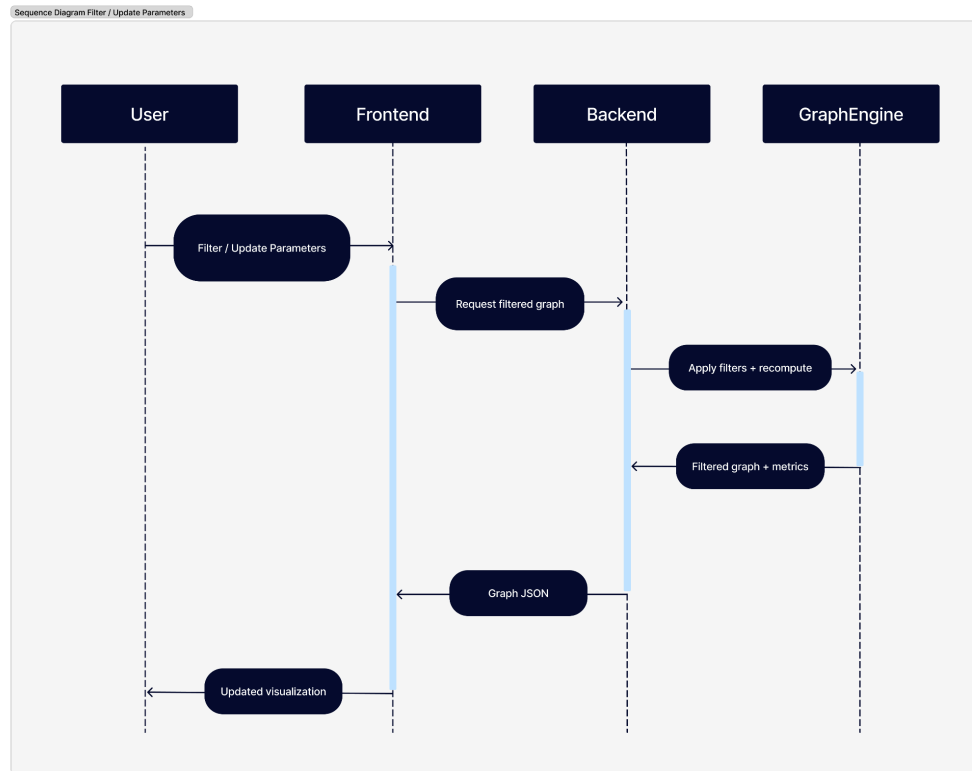


Figure 6.4.2-2 – Sequence diagram of NetXplore filter and update parameters

Sequence Diagram Compare Two Datasets

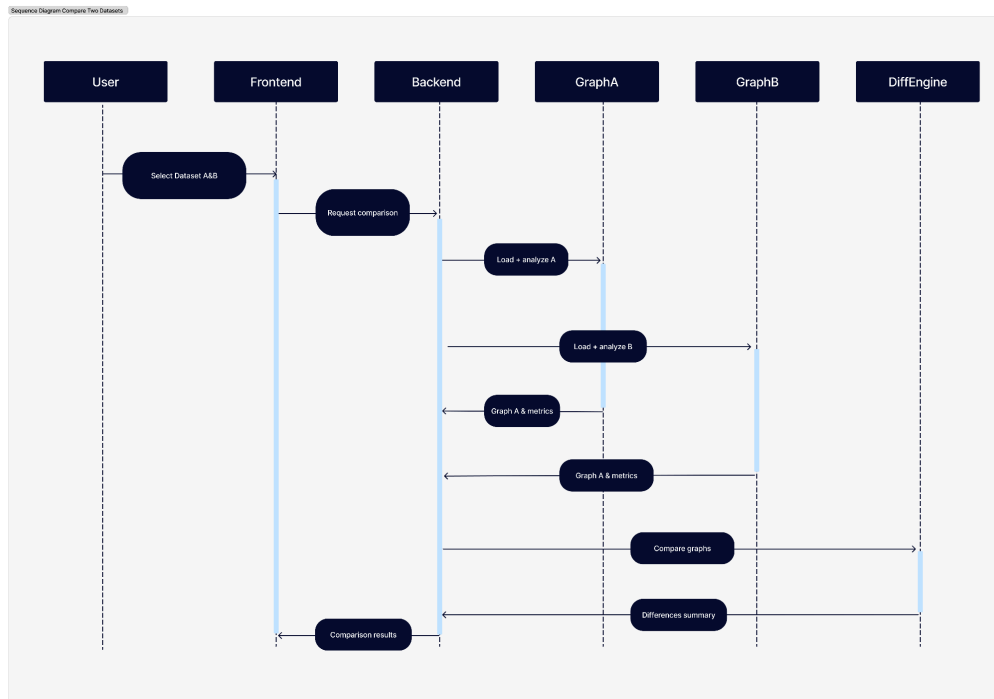


Figure 6.4.2-3 – Sequence diagram of NetXplore dataset comparison (A & B)

Sequence Diagram Export Report

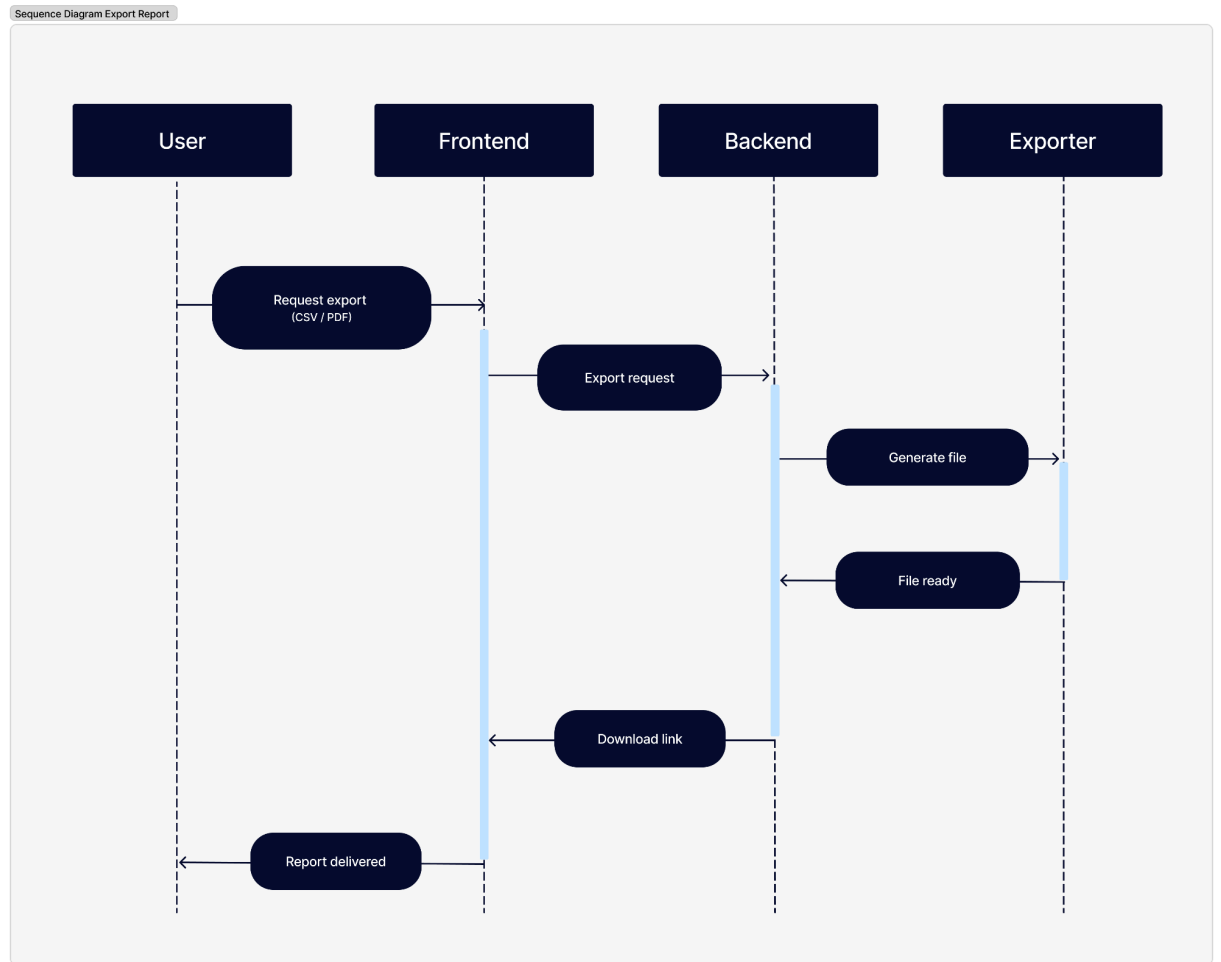


Figure 6.4.2-4 – Sequence diagram of NetXplore export report (CSV/PDF)

6.5 Software Architecture Pattern

6.5.1 Three-tier: Data, Logic and Presentation tiers

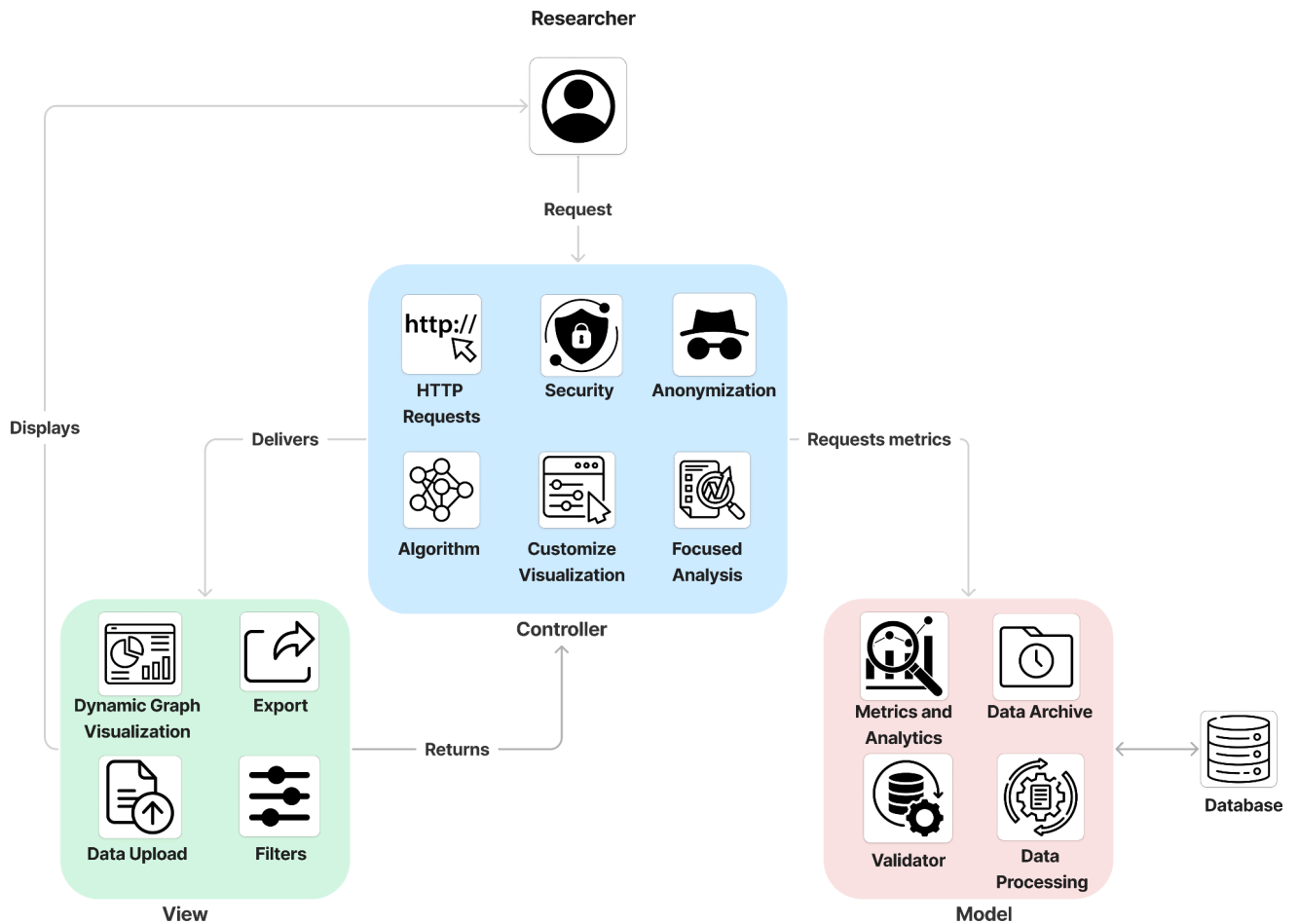


Figure 6.5.1 - Three-tier Diagram

The three-tier diagram represents the NetXplore architecture, implementing a discourse network analysis platform that processes chat and discussion data while distinctly separating the front-end presentation, backend logic, and data management layers.

- **Upload Datasets:** A front-end interface that allows users to upload WhatsApp (.txt) and Wikipedia (.json) discussion exports, which are then validated before processing into network data structures.
- **Export Researches:** A presentation-layer feature enabling users to export analysis outcomes into standardized academic formats, such as PDF and CSV, to support research documentation and external evaluation.
- **Report Researches:** A dashboard component that provides access to stored research history, allowing users to review and manage past analyses and generate formal research reports.

- **Visualization Display:** An interactive network visualization module that renders discourse networks, incorporating advanced features such as community detection and centrality metrics for enhanced interpretability.
- **Customize Visualizations:** A set of advanced filtering and customization tools, enabling users to refine visualizations by time ranges, keywords, participant activity, and message thresholds.
- **Data Processing:** A backend service layer powered by NetworkX algorithms, responsible for converting raw chat conversations into graph-based network structures.
- **Data Analysis:** The analytics engine that computes a variety of network measures, including degree, betweenness, closeness centrality, and community detection, forming the basis of social network analysis.
- **Data Filtering:** A logic-layer service providing filtering mechanisms by temporal constraints, message content, and participant activity, ensuring targeted analysis.
- **Visualization Management:** Backend processes managing graph layouts, ensuring optimal rendering performance and incorporating anonymization techniques to protect participant privacy.
- **PostgreSQL Database:** The primary data repository that persistently stores user projects, computed analysis results, and metadata, administered through pgAdmin for reliability and scalability.
- **Research Repository:** A storage and retrieval service that maintains discourse datasets and historical research records, enabling longitudinal analysis and reproducibility.

6.5.1 MVC: Model, View, Controller Structure

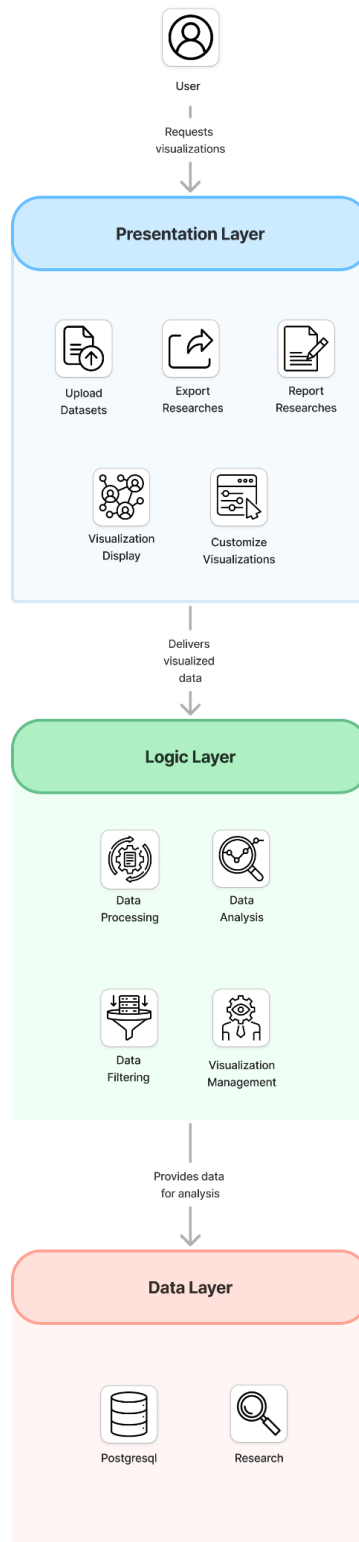


Figure 6.5.2 - Model, View, Controller Structure

The MVC architecture of NetXplore demonstrates a clear separation between data management, business logic, and user interaction. The Model layer manages data storage in PostgreSQL, dataset transformation, validation, and analytical processing, including network metrics, community detection, and historical archiving. The Controller layer is implemented with FastAPI, handling RESTful communication, authentication and security (Google OAuth and JWT), orchestration of analysis algorithms, and anonymization of sensitive data. The View layer consists of a React-based interface that enables dataset upload, advanced filtering, interactive visualization of discourse networks, focused comparative analysis, and export of results in academic formats. Together, this MVC design ensures scalability, security, and usability, supporting researchers in conducting structured and reproducible discourse network analyses.

7. IMPLEMENTATION

The implementation of NetXplore was carried out using a modular design that separates the frontend, backend, and database layers while ensuring smooth integration through well-defined APIs. The backend, developed in Python with FastAPI, handles data ingestion, preprocessing (normalization and anonymization), and analytical processes such as graph construction, centrality metrics, community detection, and the Distance Weighted Ranking (DWR) algorithm. The frontend, built with React, provides an interactive interface that enables researchers to upload datasets from WhatsApp or Wikipedia, apply filters, and explore dynamic network visualizations. All research data, configurations, and results are stored in PostgreSQL, ensuring persistence and reproducibility. This modular setup allows each component to be maintained and scaled independently, providing a flexible and environment-agnostic platform for social network analysis.

7.1 Algorithm Design

The algorithmic framework of NetXplore is designed to systematically transform raw, unstructured conversational data into structured social networks that can be rigorously analyzed both qualitatively and quantitatively. This pipeline integrates multiple stages: construction of a directed graph to preserve temporal and causal flow, weighting interactions through the Distance Weighted Ranking (DWR) method, normalization of edge weights for fair comparison, computation of centrality measures to quantify influence, and community detection to uncover group structures. Together, these stages ensure that the resulting networks capture both the chronological sequence of messages and the relational dynamics between participants.

7.1.1 Description of Algorithmic Components

NetXplore implements a set of algorithmic components that operationalize this workflow.

- **Distance Weighted Ranking (DWR):** assigns weights to interactions based on their sequential position in the conversation, emphasizing the relative influence of recent versus earlier contributions.
- **Filter-based algorithms:** allow researchers to refine datasets according to message length, time ranges, or active participants, ensuring that analyses focus on relevant subsets of data.
- **Graph analysis algorithms:** include centrality computations and community detection techniques, which identify influential nodes, structural brokers, and cohesive subgroups.

By combining these complementary components, the system ensures that raw conversation logs are transformed into analytical artifacts that are both structured and insightful.

7.1.2 Distance Weighted Ranking (DWR) Algorithm

The Distance Weighted Ranking (DWR) algorithm was developed as part of this project to model sequential conversational dynamics. Its conceptual foundation is inspired by probabilistic approaches such as Mixed-Memory Markov Processes, which capture conversational turn-taking by weighting transitions based on prior states in the sequence (Choudhury & Basu, 2004)[4]. By adopting this perspective, DWR aligns with established probabilistic methods while adapting them to the structural analysis of social discourse networks.

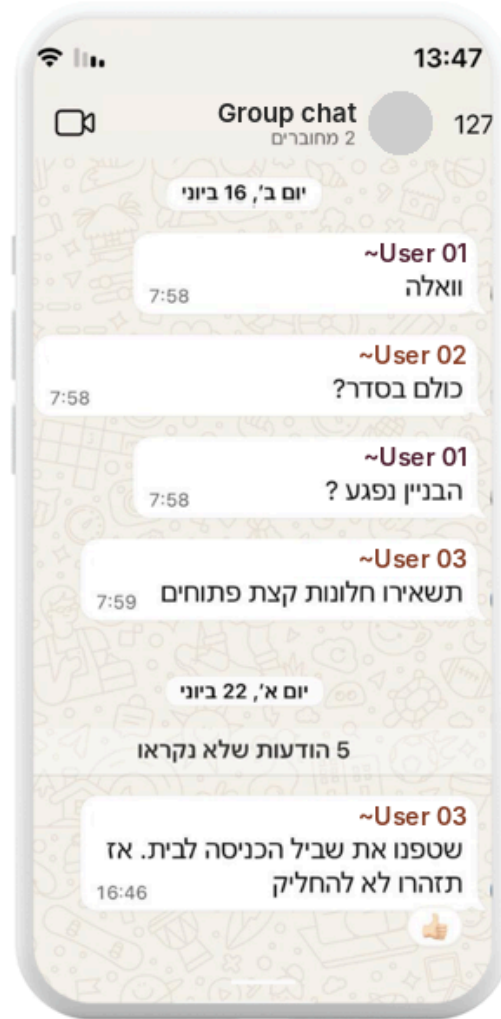


Figure 7.1.2 - 1 – Raw WhatsApp Conversation

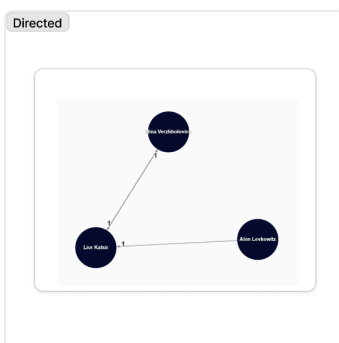
Raw WhatsApp Conversation: This figure illustrates an excerpt of a WhatsApp chat log in chronological order. While it shows who wrote what and when, it does not reveal the underlying interaction structure between participants.

NetXplore begins by constructing a directed graph from the raw conversation log. Each message is treated as a node originating from its sender, and it is linked only to the message(s) it directly replies to. In this initial direct reply graph, every such connection is

represented with an edge of weight 1. This produces the most basic interaction network, essentially the linear or hierarchical thread structure of the conversation without any additional weighting. The directed nature of the graph is critical: without directionality, the temporal flow and causal structure of communication would be lost. By enforcing edges from the current message's sender to the previous message's sender, the system encodes the order in which information is exchanged.

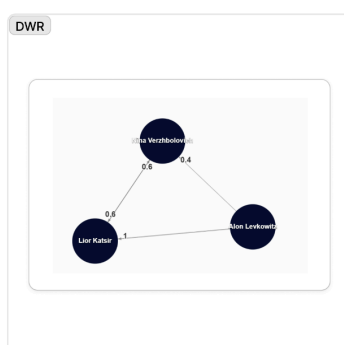
This process of inferring reply connections can differ between platforms. For WhatsApp, replies are often implicit in the sequence of messages, so the system uses temporal ordering to infer which prior message a new message is responding to. In Wikipedia talk pages (or similar threaded forums), where discussions are nested hierarchically, the parser respects indentation levels or quoting conventions to determine each reply's target. Regardless of the platform, the outcome is a directed graph that serves as the structural backbone for all subsequent analysis.

After building this directed graph of direct replies, additional computational layers are applied to enrich the representation of the conversation. In summary, the conversation is transformed through several stages (beyond the basic direct-reply graph):



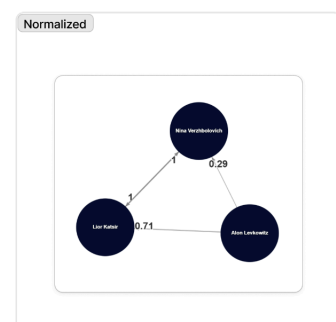
Directed – directed graph

Each message is linked only to the one it directly replies to, with every edge assigned weight 1. This produces the most basic structure, representing the raw linear or hierarchical flow of the conversation.



DWR – graph with distance-based weights

The Distance Weighted Ranking (DWR) algorithm extends the graph by connecting each message not only to its immediate predecessor but also to a configurable number of earlier messages (n_{prev}). Each step back is given a decreasing weight (e.g., 0.6, 0.4), calculated by a utility function that aggregates contributions across the sequence. This highlights both direct replies and the residual influence of earlier messages.



Normalized – normalized graph

To allow fair comparisons between participants, edge weights are normalized so that the total incoming weights to each node sum to 1. This emphasizes the *relative* share of influence each participant has, rather than absolute message counts.

Important clarification: The DWR algorithm operates only on a directed graph. It is based on the chronological order of messages, so it creates directed edges from the current sender to previous senders and assigns weights according to the time distance. If you work with an undirected graph, this directionality is lost and the calculated weights cease to have meaning.

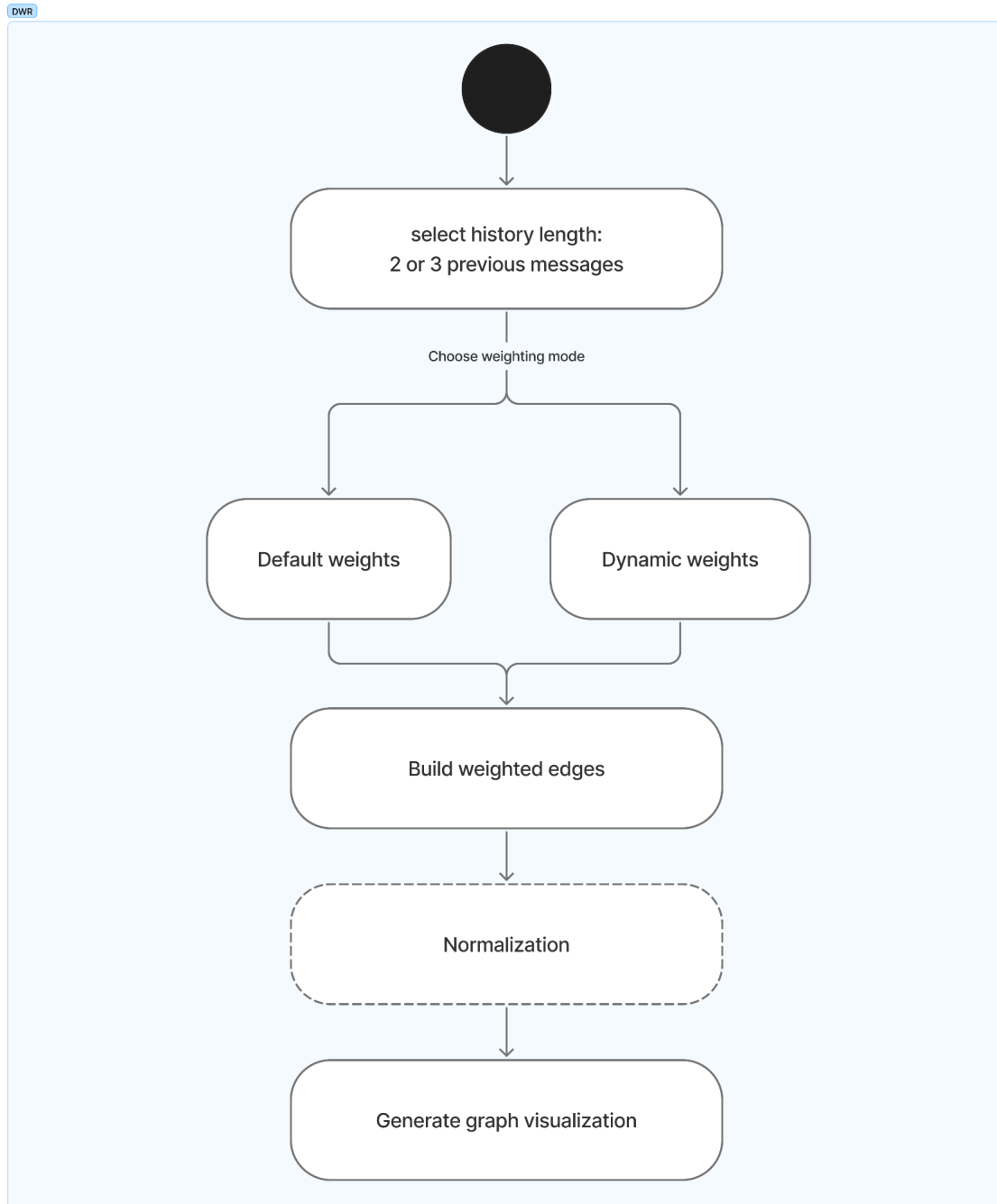


Figure 7.1.2 - 2 – DWR Workflow Diagram

Directed Graph Construction

At the foundation of the system lies the construction of a directed graph, which models conversational exchanges as interactions between participants. Each message is treated as originating from the sender and pointing back to the authors of previous messages. The directed nature of the graph is critical: without directionality, the temporal flow and causal structure of communication would be lost. By enforcing edge direction from the current sender $\rightarrow$ previous sender(s), the system encodes the order in which information is exchanged. This process differs between platforms. For WhatsApp, replies are often implicit in the sequence of messages, so the system uses temporal ordering to infer connections. For Wikipedia talk pages, where threads are nested hierarchically, the parser respects indentation and quoting conventions to determine the target of each reply. Regardless of the platform, the graph remains directed and forms the structural backbone for all subsequent analysis.

Normalization of Weighted Edges

After applying DWR, the graph contains edges with varying absolute weights. However, raw weights may be misleading when comparing participants with different levels of activity. To address this, *NetXplore* employs a normalization procedure.

The function `normalize_links_by_target` ensures that all incoming edges to a node sum to 1. In other words, the weights are transformed into relative shares of influence. For example, if one participant contributes edges of weight 0.71 and another of weight 0.29, the normalized graph reveals the proportional impact of each contributor on the same target.

This normalization makes it possible to compare influence fairly across nodes, independent of their absolute activity volume. It is particularly useful in heterogeneous datasets, where some users dominate with frequency while others exert influence through fewer but strategically placed messages.

Centrality Measures

With the directed, weighted, and normalized graph constructed, *NetXplore* applies a series of centrality algorithms to measure node importance from multiple perspectives. These include:

- **Degree Centrality** – measures the fraction of direct connections a node has:

$$C_D(v) = \frac{\deg(v)}{n-1}$$

- **Closeness Centrality** – captures how close a node is to all others in terms of path distance:

$$C_C(v) = \frac{n-1}{\sum_{t \neq v} d(v,t)}$$

- **Betweenness Centrality** – identifies nodes that act as bridges on shortest paths:

$$C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

- **Eigenvector Centrality** – extends degree by weighting connections to already influential nodes:

$$A \cdot x = \lambda x$$

- **PageRank Centrality** – distributes importance recursively across links:

$$PR(v) = \frac{1-d}{N} + d \sum_{u \in M(v)} \frac{PR(u)}{L(u)}$$

Together, these measures provide a multi-dimensional view of influence, identifying not only the most active participants but also those who are strategically positioned to mediate, amplify, or bridge conversations.

Community Detection

Finally, NetXplore applies community detection algorithms to uncover the hidden group structures within the network. Two complementary approaches are supported:

- The **Louvain algorithm**, which optimizes modularity by aggregating nodes into communities and progressively building hierarchical clusters. This method is highly efficient and well-suited for large datasets, producing compact and interpretable communities.
- The **Girvan–Newman algorithm**, which identifies communities by iteratively removing edges with high betweenness centrality. This approach provides an alternative lens, often revealing hierarchical splits and cross-group bridges.

The results of community detection enrich the analysis by showing how discussions fragment into subgroups, which participants connect otherwise separate communities, and whether information flows primarily within clusters or across them.

7.2 Interfaces

This section will present the user interface, which is the web-based application. The section will include screenshots of the different screens of the GUI.

7.2.1 NetXplore Welcome Page.

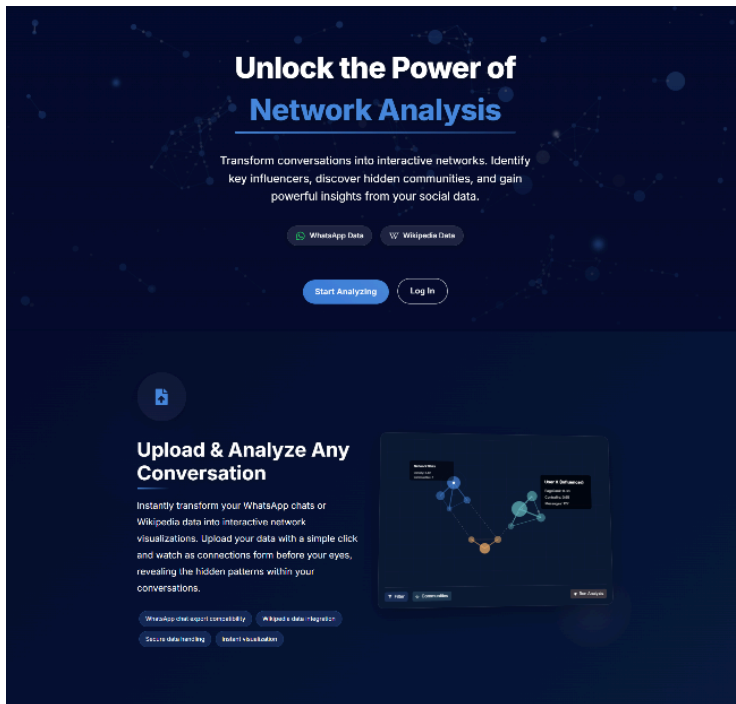
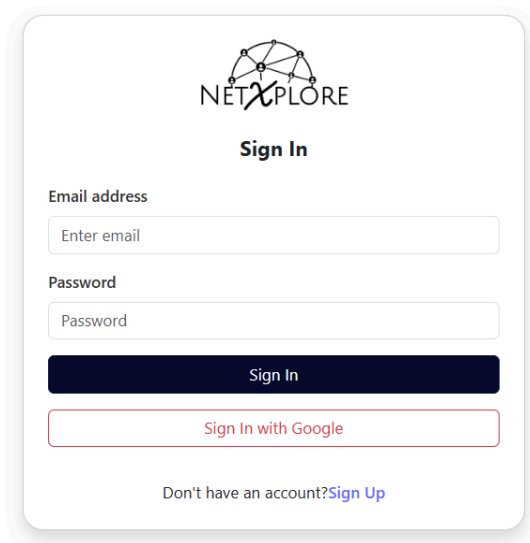


Figure 7.2.1- Welcome Screen Page

The entry screen presents the platform's vision, highlighting network analysis capabilities and offering users to log in or start analyzing conversations.

7.2.2 Login Page



The image shows a 'Sign In' screen for NetXplore. At the top is the NetXplore logo, which consists of a network diagram with nodes and lines, and the text 'NETXPLORE' below it. Below the logo is the title 'Sign In'. There are two input fields: 'Email address' with a placeholder 'Enter email' and 'Password' with a placeholder 'Password'. Below these fields is a dark blue 'Sign In' button. Underneath the button is a red-outlined button that says 'Sign In with Google'. At the bottom, there is a link that says 'Don't have an account? Sign Up'.

Figure 7.2.2 - Sign In Screen Page

The login interface of *NetXplore*, allowing users to sign in with email and password or via Google authentication.

7.2.3 Home Dashboard

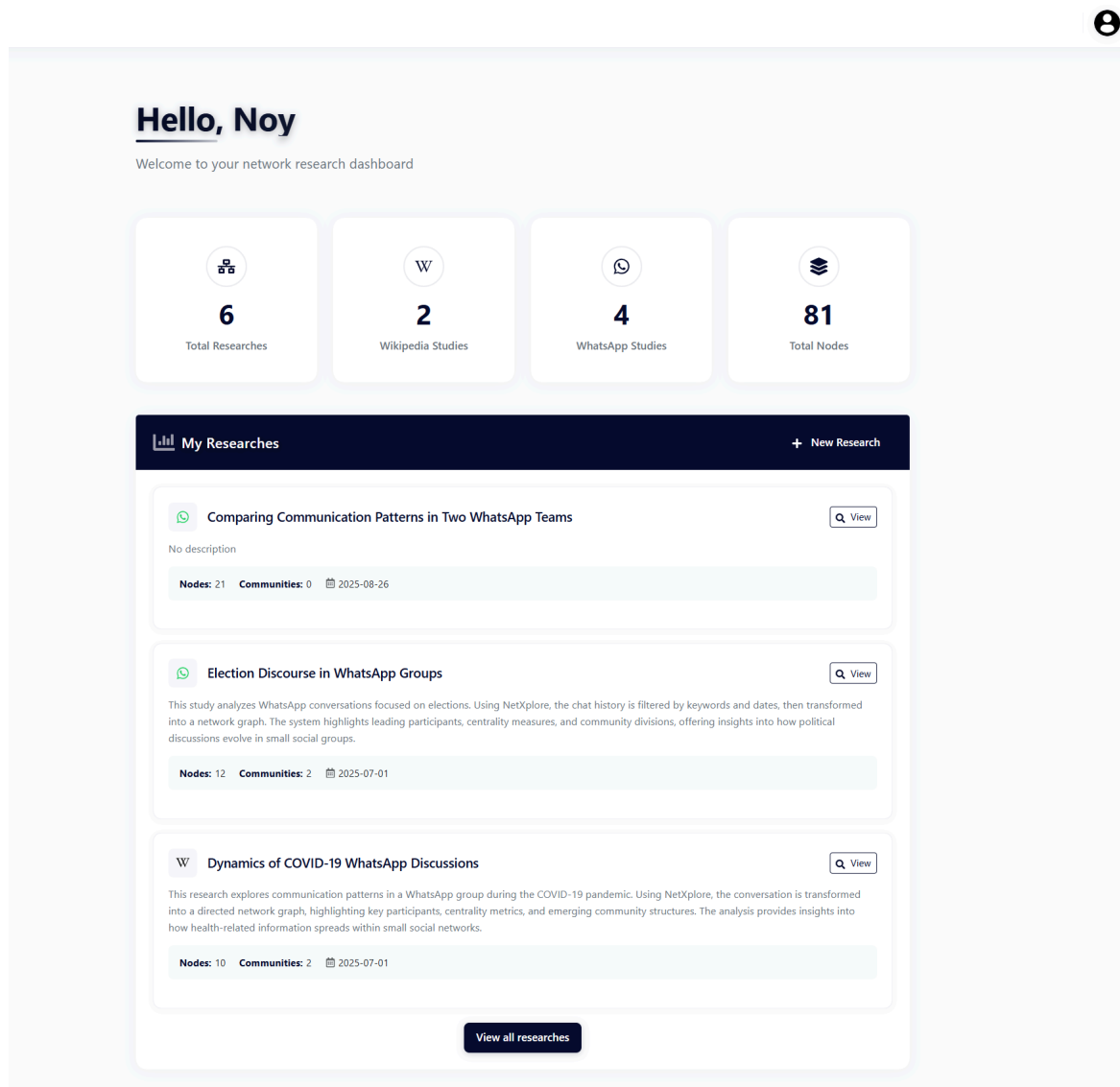


Figure 7.2.3- Home Dashboard Screen Page

The home dashboard provides an overview of the user's research activity. It displays summary statistics such as the total number of researches, studies per platform (WhatsApp and Wikipedia), and the total number of nodes analyzed. Below the summary, the My Researches section lists individual projects, allowing users to view details or start new analyses.

7.2.4 Research History

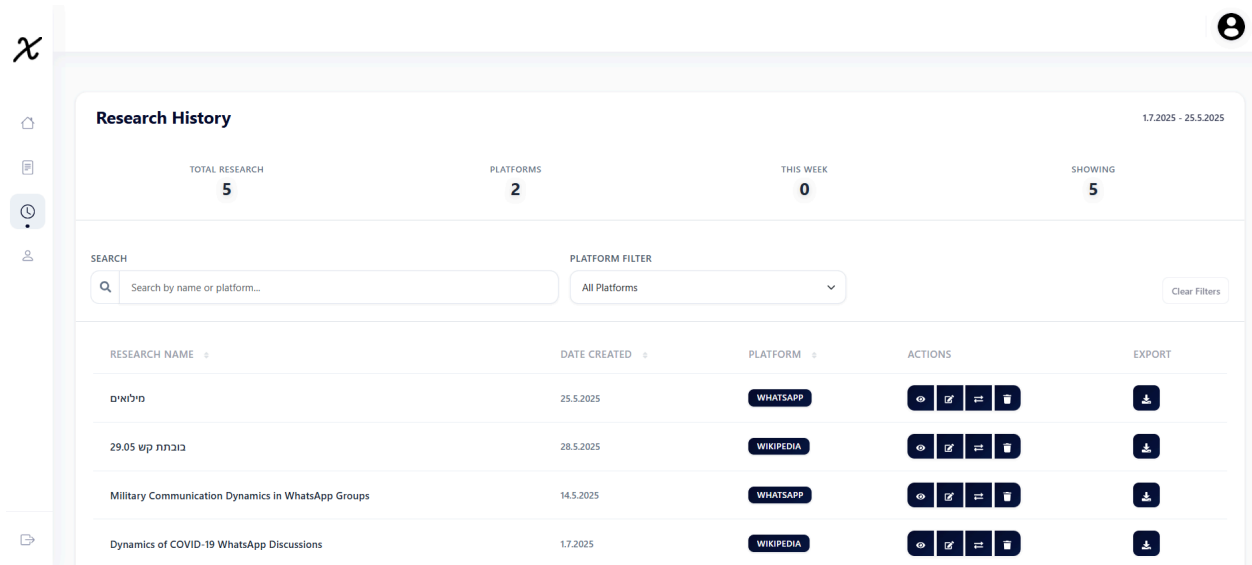


Figure 7.2.4- Research History Screen Page

Displays all previous studies with options to view, edit, delete, export to CSV, and compare results.

7.2.5 Platform Selection Page

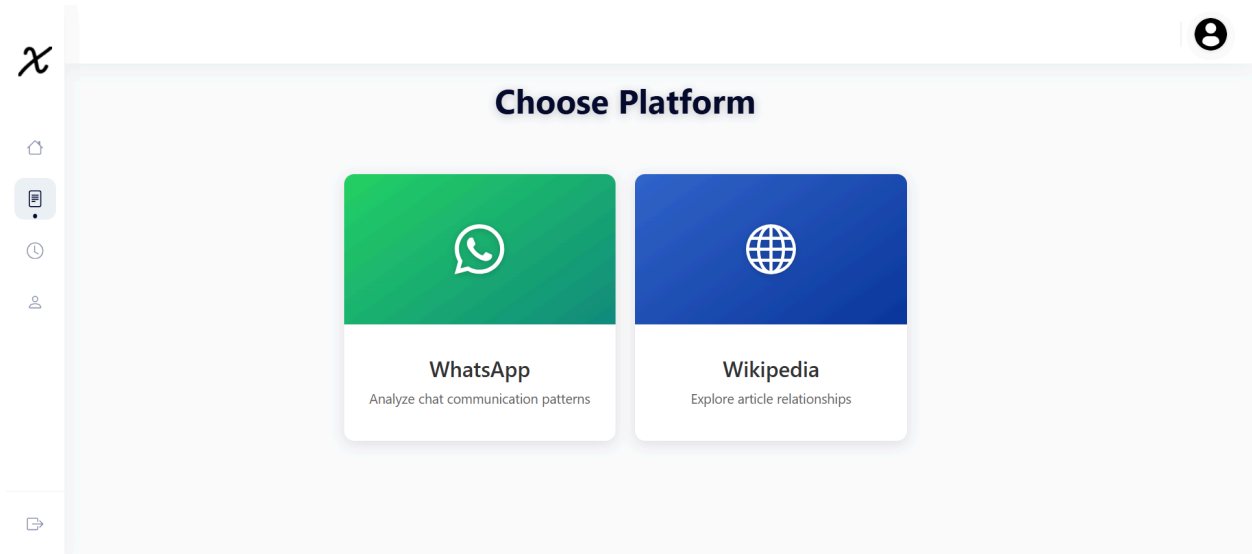


Figure 7.2.5- Platform Selection Screen Page

The platform selection page, where the user chooses between analyzing WhatsApp or Wikipedia data.

7.2.6 Research Setup Wikipedia

Network Research
Configure your network analysis in a few simple steps

1 Setup 2 Discussion 3 Config 4 Time 5 Content 6 Users 7 Network 8 Compare 9 Report

Research Setup

Research Name*
Enter a name for your research
A descriptive name to identify this research

Description
Describe the purpose of this research
Optional description of your research goals

Wikipedia Discussion URL*
Paste a Wikipedia discussion link
This will fetch and analyze discussion data from Wikipedia.

Upload URL

< Previous Next >

Figure 7.2.6- Research Setup Wikipedia Screen Page

If the Wikipedia option is selected, the setup step requires the researcher to provide a discussion URL. The system fetches and analyzes the selected Wikipedia talk page, transforming it into structured data for network analysis.

7.2.7 Discussion page

Network Research
Configure your network analysis in a few simple steps

1 Setup 2 Discussion 3 Config 4 Time 5 Content 6 Users 7 Network 8 Compare 9 Report

Discussion Sections

Found: 42 sections with 407 comments

Search by title... Most comments ▾

קבוצת הוואצאפ של ראשי המחאה 41 Comments	הצעה לפסקה: הישגי המחאה 40 Comments	תקיפת כתבת ערוץ 14 38 Comments	מצעד התמיכה בישראל שהתקיים במנהטן 33 Comments
אלימות המשטרה 28 Comments	שיטות הפסקת התנגדות מאורגנת 22 Comments	פיצול הערך 18 Comments	המחאה במהלך המלחמה 18 Comments
מחיקת מידע על מחאת הסטודנטים 17 Comments	הפסקת התרומות של דנצי'ק 13 Comments	מחאת הקהילות בחו"ל 12 Comments	עדכון אירועים, הצהרת נתיחה, פרישת אנשי חיל אוויר 11 Comments
הקשר בין המחאות 11 Comments	דריסה מפגנים שזולמה בשידור חי 10 Comments	הניסיון לפטר את גלנט במרץ 2023 9 Comments	מחיקת פסקה שעסקה באירועי אלימות מה-24 ליולי 8 Comments
דברי עמי אשד עם פרישתו 6 Comments	שינוי שם 6 Comments	שינוי שם 6 Comments	חוק קרן הארנונה 6 Comments
דיונים סותרים ממניעים זרים 5 Comments	מה המקור למספר הפצועים? 5 Comments	יצירת חשבונות און ליין המתחזים לפעילי המחאה על ידי מדינה זרה 4 Comments	המחאות בארה"ב במהלך ביקור נתיניה בארצות הברית 4 Comments
אין טייסים להטיס את ראש הממשלה 4 Comments	איחוד ערכים 3 Comments	מחאה נגד השינויים במערכת המשפט 3 Comments	"חצי ממשלה" או "כלל השרים" 3 Comments
הסתה נגד המחאה מאז פרוץ המלחמה 3 Comments	הרשעה בניסיון דריסת מפגנים 2 Comments	תקיפת מפגין על ידי שוטרי 2 Comments	אלימות המפגנים 2 Comments
פתיחת חקירת מח"ש נגד מפקד יס"מ תל אביב על אלימות כלפי מפגנים 2 Comments	משוב מ-27 ביולי 2023 2 Comments	שלילת רשיון ממפגנים 2 Comments	טענה למכתב של 10,000 חיילי מילואים 2 Comments
פרק על תקרית אלימות באירועי המחאה 1 Comments	נמצאו קישורים חיצוניים שצריכים תיקון (ינוי 2023) 1 Comments	בעיה בדיווח המשטרה על אלימות מפגנים 1 Comments	לסגור את הדף לעריכה 1 Comments
בעיה בקישורי ארכיון בערך (אוקטובר 2023) 1 Comments	שביתת מועצת התלמידים לא שייך "תחת" "אקדמאים" 1 Comments		

< Previous Next >

Figure 7.2.7- Discussion Screen Page

When the Wikipedia option is selected, the system fetches the linked discussion page and displays all available sections. The researcher can then choose a specific section to analyze, based on title or number of comments

7.2.8 Research Setup WhatsApp

Network Research

Configure your network analysis in a few simple steps

1

Setup

2

Config

3

Time

4

Content

5

Users

6

Network

7

Compare

8

Report

Research Setup

Research Name*

Enter a name for your research

A descriptive name to identify this research

Description

Describe the purpose of this research

Optional description of your research goals

Upload Data File*

Choose File

No file selected

Upload a WhatsApp chat export (.txt)

Note: Media (images, videos, audio, GIFs) is not included. Only text messages from the exported chat will be analyzed. [How to export a WhatsApp chat?](#)

< Previous

Next >

Figure 7.2.8- Research Setup WhatsApp Screen Page

If the WhatsApp option is selected, the setup step requires the researcher to upload a conversation file in `.txt` format. The system processes the raw chat history, cleans and structures it, and then transforms it into a network representation ready for analysis.

7.2.9 Configuration Page

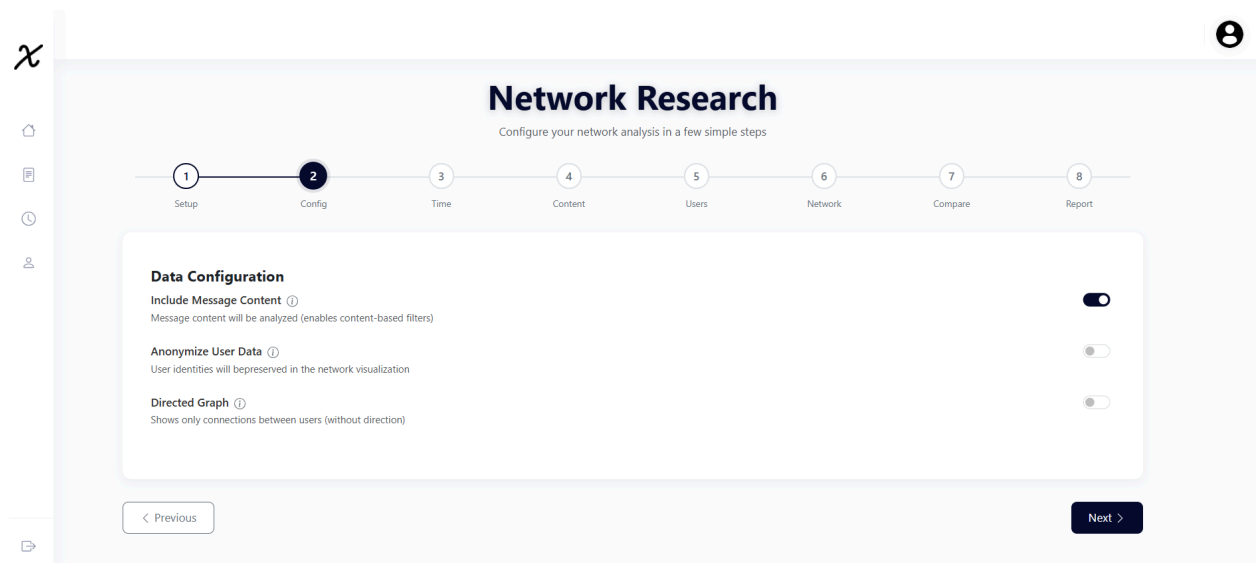


Figure 7.2.9 - Configuration Page Screen Page

On this page, the researcher configures how the data will be processed before building the graph. Three main options are available:

- **Include Message Content** – Enables the system to analyze the actual text of the messages, allowing the use of content-based filters.
- **Anonymize User Data** – Ensures user identities remain hidden, presenting participants anonymously in the network visualization.
- **Directed Graph** – Preserves the direction of interactions (who replied to whom). If disabled, the system produces an undirected graph that only reflects mutual connections.

This step is crucial as it allows the researcher to determine whether the analysis should focus solely on the network's structure, also incorporate the content of the discourse, and whether user privacy should be preserved.

Network Research

Configure your network analysis in a few simple steps

1

Setup

2

Config

3

Time

4

Network

5

Compare

6

Report

Data Configuration

Include Message Content ⓘ

Only communication patterns will be analyzed (no content analysis)

☐

Anonymize User Data ⓘ

User identities will be preserved in the network visualization

☐

Directed Graph ⓘ

Shows direction of communication (who messaged whom)

☒

Use Distance Weighted Ranking Algorithm ⓘ

Include temporal patterns in the analysis

Message History Length ⓘ

3 messages

Number of previous messages to include in the analysis

Previous Messages Weights ⓘ

Sum: 1.0

Message 1 (most recent)

0.1 0.5 1.0

First weight - changes affect all others proportionally

Message 2 (2nd recent)

0.1 0.3 1.0

Changing this weight preserves the first weight

Message 3 (3rd recent)

0.1 0.2 1.0

Changing this weight preserves the first weight

Normalized Algorithm ⓘ

Adjust for differences in user activity levels

☐

< Previous

Next >

The configuration page allows researchers to customize how the conversation data will be analyzed. Options include whether to analyze message content, anonymize user data, and use a directed graph. Advanced settings enable applying the Distance Weighted Ranking (DWR) algorithm by selecting how many previous messages to include and assigning weights to each. A normalization option can also be applied to adjust for differences in user activity levels.

7.2.10 Time Frame Page

The screenshot shows the 'Time Frame' configuration page in the NetXplore application. At the top, a progress bar indicates seven steps: 1. Setup, 2. Config, 3. Time (current step), 4. Users, 5. Network, 6. Compare, and 7. Report. The main title 'Network Research' is centered, with the subtitle 'Configure your network analysis in a few simple steps' below it. The 'Time Frame' section contains several input fields: 'Start Date (Optional)' and 'End Date (Optional)' with date pickers and 'dd/mm/yyyy' placeholders; 'Start Time (Optional)' and 'End Time (Optional)' with time pickers and '--:--' placeholders. A checkbox labeled 'Limit Data' is checked, with a sub-label 'Restrict analysis to a specific number of messages'. Below this is a 'Limit Count' input field with the value '50'. To the right, a 'Direction' section has two radio buttons: 'From Start' (selected) and 'From End'. At the bottom, there are '< Previous' and 'Next >' navigation buttons.

Figure 7.2.10 - Time Frame Screen Page

The Time Frame page allows the researcher to define start and end dates, as well as optional times, to limit the analysis to a specific period. Additionally, a limit can be set on the number of messages included, either from the start or from the end of the dataset.

7.2.11 User Filters Page

Network Research

Configure your network analysis in a few simple steps

1

Setup

2

Config

3

Time

4

Users

5

Network

6

Compare

7

Report

User Filters

Minimum Messages Per User (Optional)

1

Only include users who sent at least this many messages

Maximum Messages Per User (Optional)

Only include users who sent at most this many messages

Active Users Threshold (Optional)

0

Filter to only show users with at least this many connections (0 to show all)

Username Filter (Optional)

Enter username to filter

Only include users whose names contain this text

< Previous

Next >

Figure 7.2.11 - User Filters Page

This page allows researchers to refine their analysis by applying user-based filters. Options include setting a minimum or maximum number of messages per user, defining an activity threshold to highlight only highly connected participants, and filtering by specific usernames. These filters help focus the analysis on the most relevant or active users within the dataset.

7.2.12 Network Analysis Page

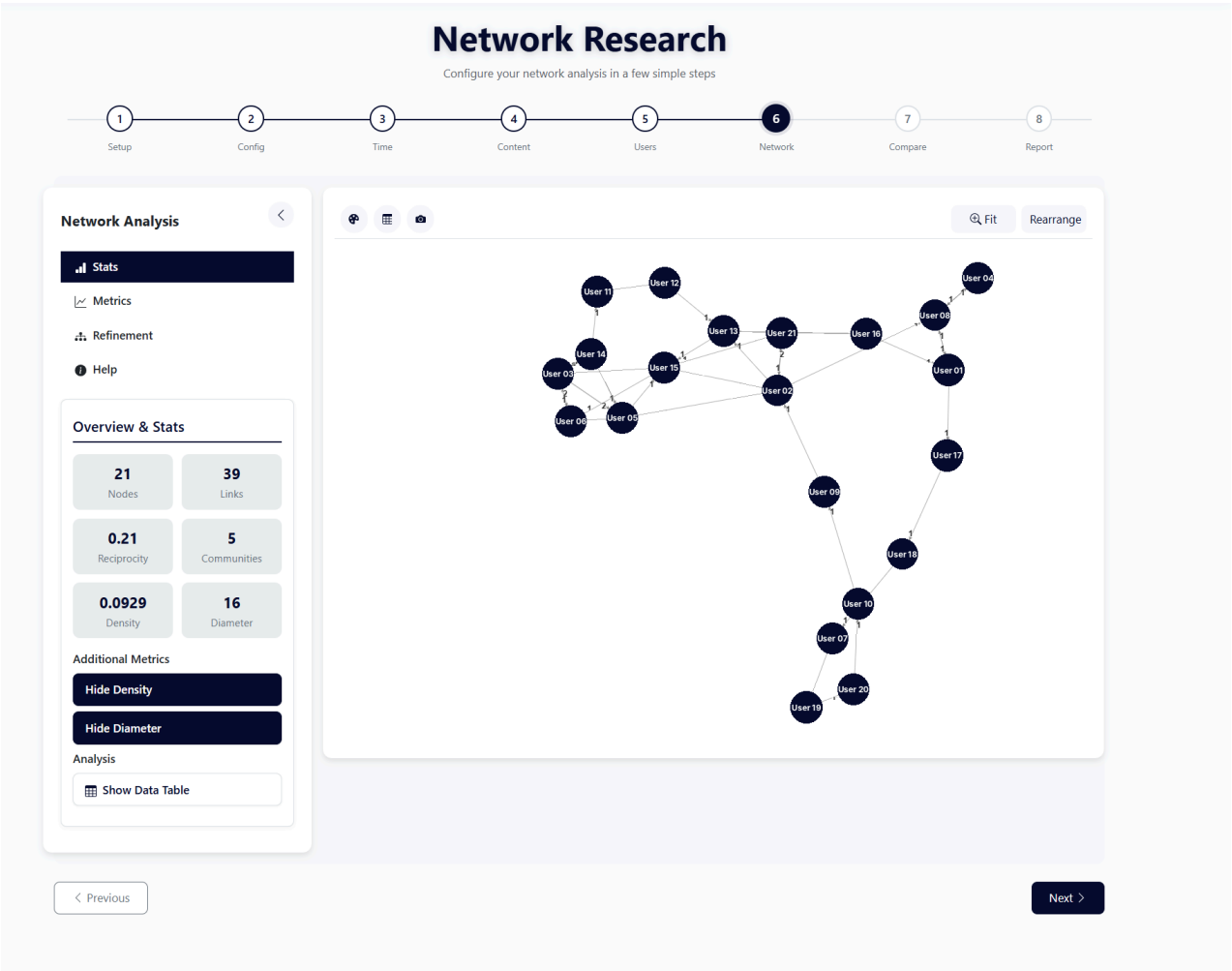


Figure 7.2.12 - Network Analysis Page

This page displays the network visualization, showing nodes (participants), links (interactions), and key metrics such as density, reciprocity, diameter, and community count, allowing researchers to explore the overall structure of the conversation.

7.2.13 Network Data Table

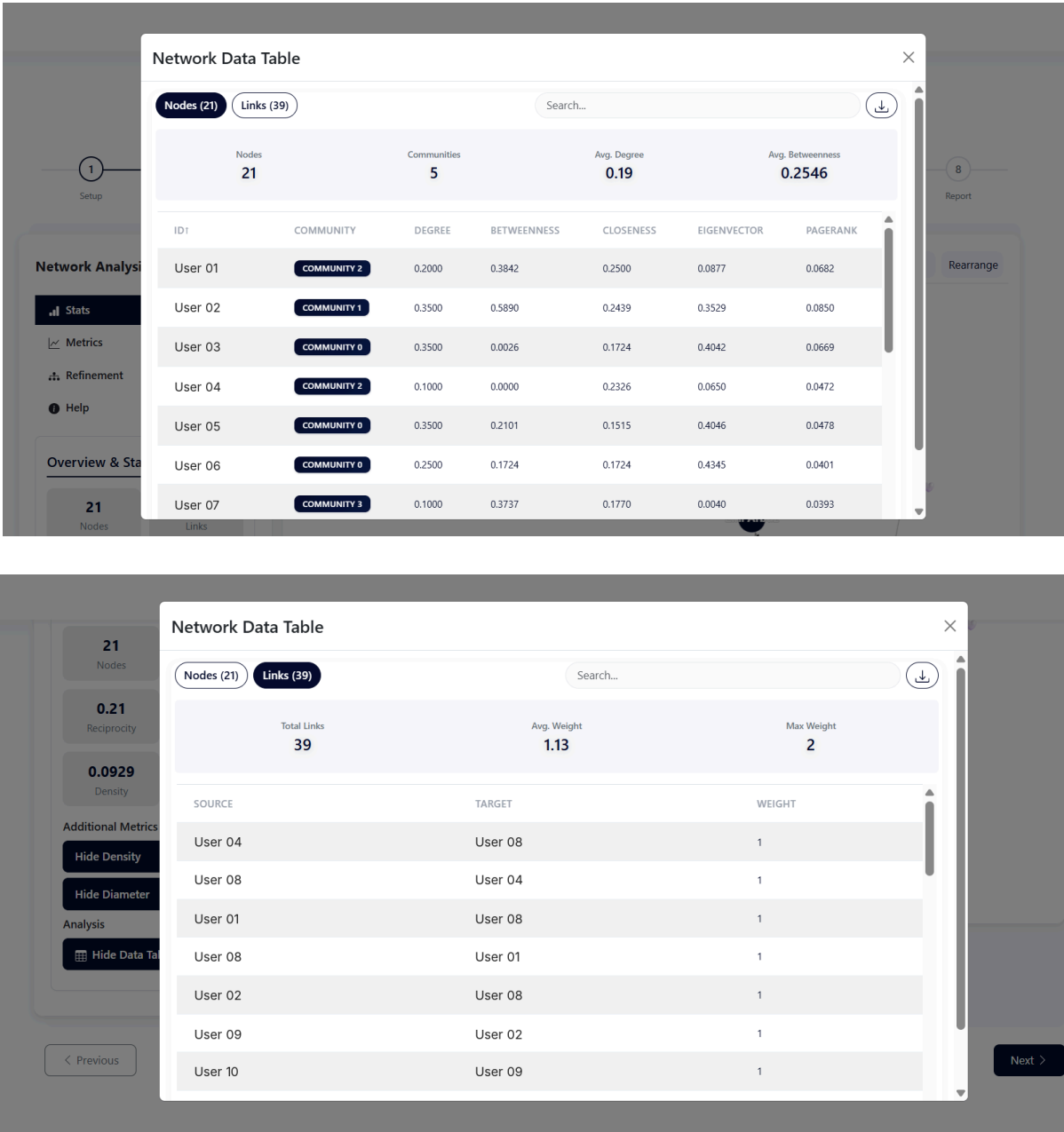


Figure 7.2.13 -Network Data Table

The Network Analysis interface presents researchers with a clear and user-friendly view of social interactions. The main graph illustrates connections between participants, where nodes represent users and links represent their interactions, enabling an intuitive understanding of the overall structure. Alongside the visualization, the system provides detailed data tables: one that lists all links, showing who interacts with whom and with what weight, and another that displays node-level information such as community affiliation and centrality measures. This combination of visual and tabular perspectives allows researchers to easily explore both the global network patterns and the role of individual participants without overwhelming repetition or complexity.

7.2.14 Visualization Customization

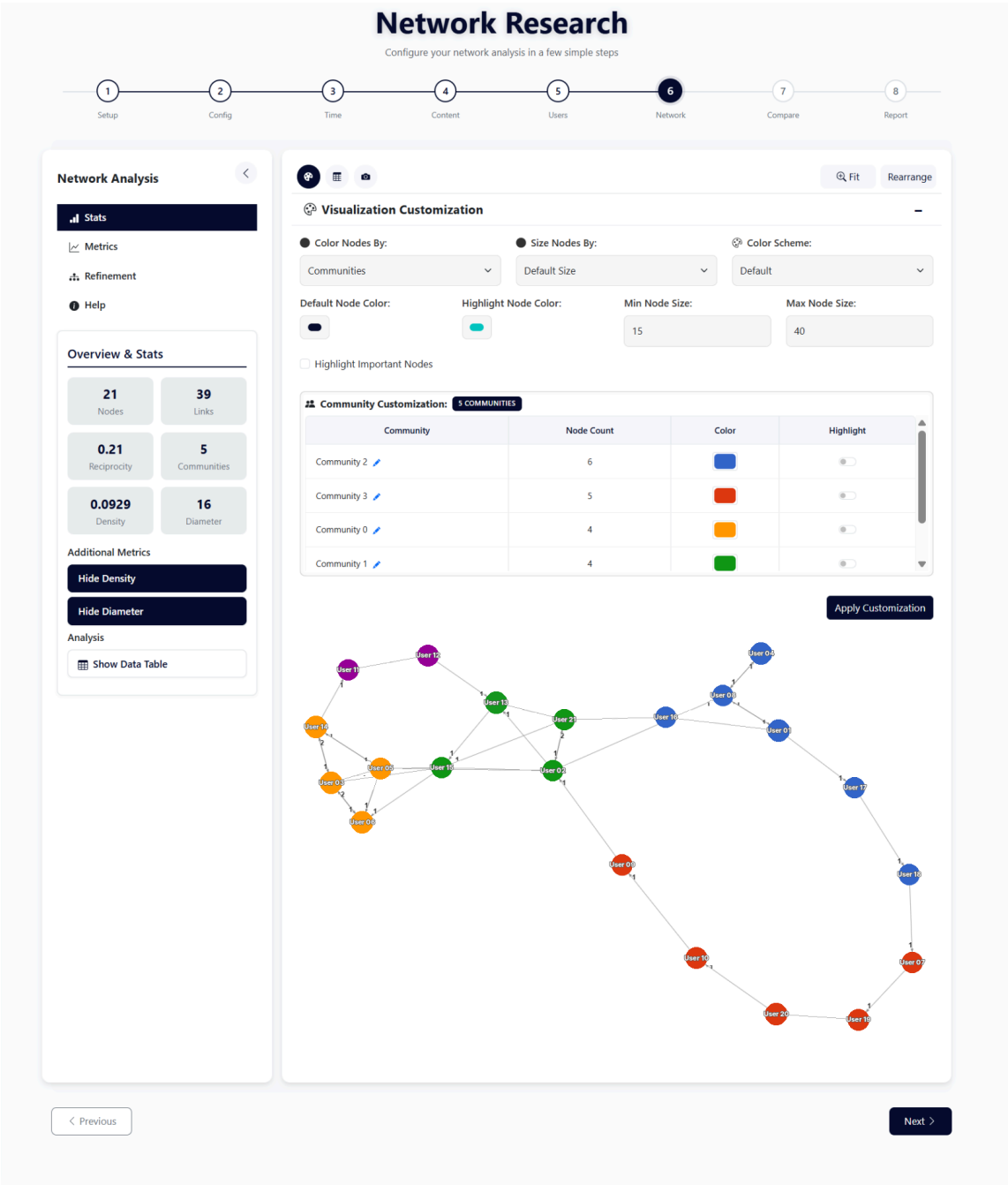


Figure 7.2.14 - Visualization Customization

The visualization customization page allows researchers to personalize the graph by adjusting node size, color, and highlighting options. Communities are automatically detected and can be color-coded for clarity, helping to distinguish different groups and patterns within the network.

7.2.15 Comparison Page

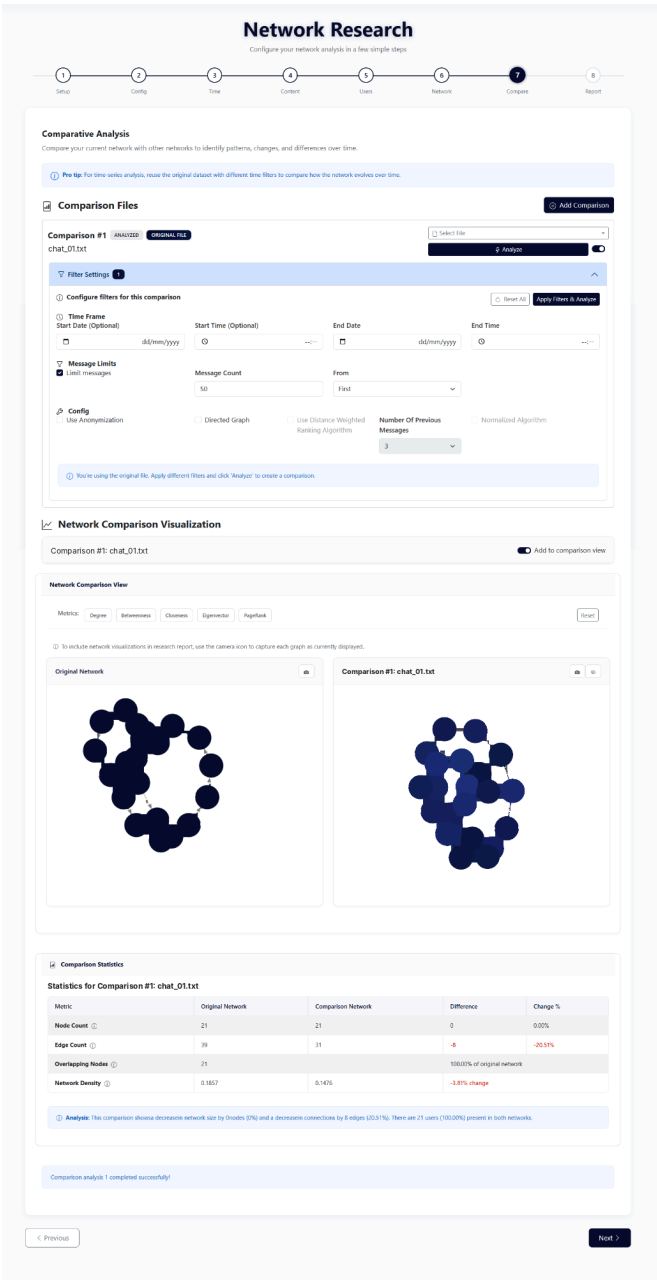


Figure 7.2.15 - Comparison Page

The comparison page allows researchers to perform multiple comparisons between networks. Users can either upload a new dataset to compare against the current one, or reuse the same dataset with different filters (such as time range, message limits, or user constraints). This flexibility enables both cross-dataset analysis where two completely different conversations are compared and intra-dataset analysis, where the same conversation is examined under varying conditions. By supporting multiple comparisons, the system provides deeper insights into how communication patterns evolve, highlighting structural changes, shifts in centrality, or variations in community formation across different contexts.

7.2.16 Research Report Page

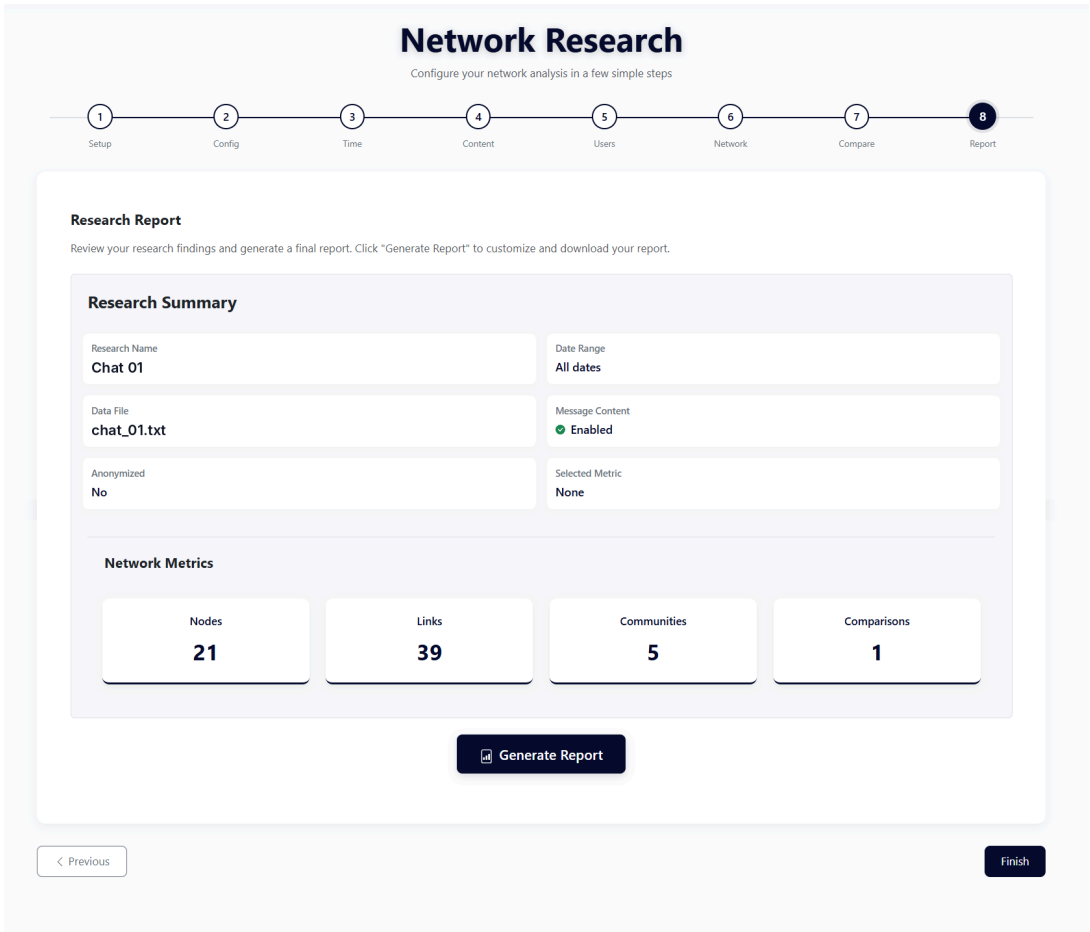


Figure 7.2.16 - Research Report Page

The final step of the workflow provides a comprehensive summary of the research configuration and results. It displays key details such as the research name, selected dataset, date range, whether message content was analyzed, and anonymization status. In addition, network metrics including the number of nodes, links, communities, and comparisons are clearly presented. Researchers can review these details and then generate a downloadable report, ensuring the analysis is preserved and easily shared.

7.2.17 Report Generation Popup

Research Report [X]

Research Information

Research Name: Chat 01

Researcher Name: Noy

Include Images

Main Image

 ☒ Include this image

Image description

Source Image

 ☒ Include this image

Image description

☒ Message Weights ☒ Is Normalized ☒ Min Length ☒ Min Messages

☒ Message Weights ☒ Is Normalized ☒ Min Length ☒ Min Messages

Research Conclusion. (is required)

☒ Include Comparison

Cancel **Download Report**

Figure 7.2.17 - Report Generation Popup

The Report Generation page enables researchers to finalize and customize their study output before download. Alongside displaying key details such as research name, data file, metrics, and filters, the page also offers options to add custom text, researcher information, and conclusions. Researchers can include comparative results and selectively display applied configurations (e.g., message weights, normalization, thresholds). In addition, images and illustrations can be attached to the report if the researcher has chosen to capture network screenshots, ensuring visual evidence of the analysis is embedded in the final document.

7.2.18 Save Research Pop-Up

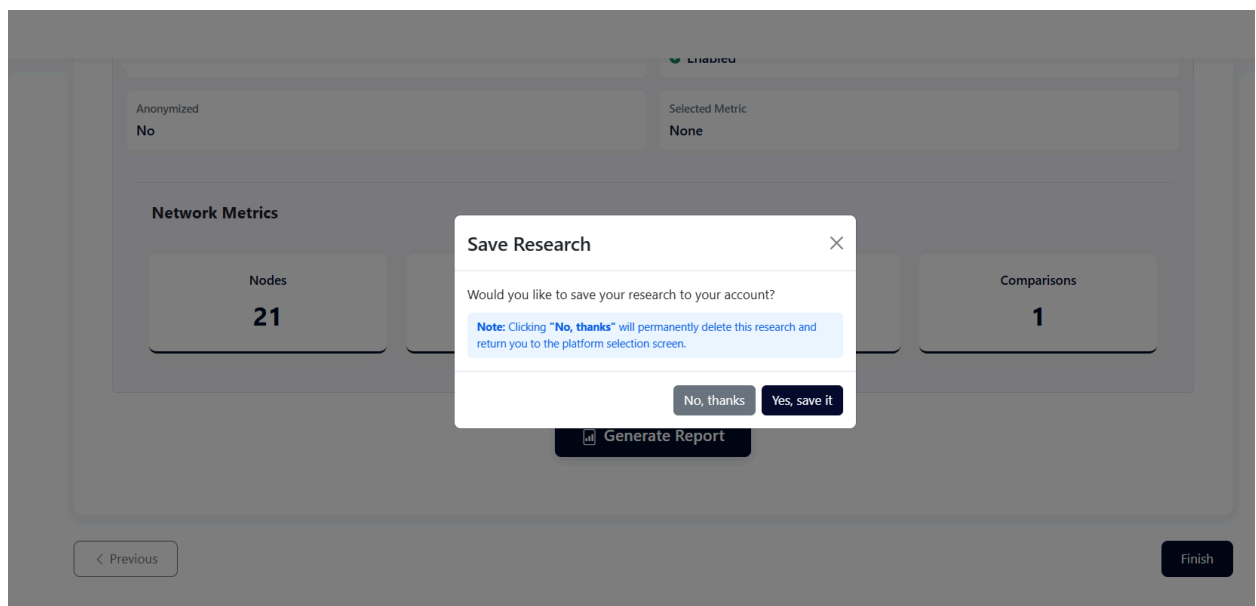


Figure 7.2.4 - Save Research Pop-Up

This pop-up ensures that researchers have full control over saving their studies. By offering the option to either save or discard the research, the system prevents unnecessary data retention and safeguards user privacy. Choosing “No, thanks” immediately deletes the research, ensuring sensitive data is not stored. This design highlights the platform’s commitment to data security and privacy protection, keeping user information secure and only preserved when explicitly authorized.

[Link to Full Report](#)

7.3 Development environment

NetXplore is implemented as a modern web application with a clear separation between the client and server layers. The platform’s workflow spans data ingestion, parsing, graph construction and interactive analysis, all orchestrated through the following environment:

Frontend: Built with React (Vite-based build system) to deliver a responsive user interface. It uses JSX components, React Hooks for state management, and libraries such as react-force-graph for network visualization and react-table for tabular views. The UI is optimized for desktop and tablet screens, supports internationalization, and includes accessibility features.

Backend: Implemented in Python using the FastAPI framework. FastAPI’s asynchronous capabilities enable high-throughput endpoints for dataset upload, graph construction, metric calculation and report generation. The backend modules include WhatsApp and Wikipedia parsers, a preprocessing layer that handles normalization and optional anonymization, the DWR algorithm for weighted edges, and a graph-analysis layer based on NetworkX and community-detection algorithms (e.g., Louvain). Responses are serialized as JSON and consumed by the React front-end.

Database & Storage: Interaction data, computed graphs, user configurations and saved research projects are stored in a PostgreSQL database managed through Xata. Temporary files and exports (CSV/PDF) are stored in an object store (e.g., a cloud bucket) with signed URLs for download. The system supports session-based processing to ensure that unsaved data is purged automatically after analysis.

Infrastructure & Tooling: The project uses Docker containers for reproducible development and deployment. A docker-compose script starts the frontend, backend and a local Postgres/Xata emulator. Continuous integration runs unit tests and linters on pull requests. In production, services can be deployed to a cloud provider (e.g., a virtual machine or container service) with HTTPS termination and environment-specific configurations.

7.4 Programming languages

Frontend

JavaScript(React + Vite) – responsible for all user-facing components: page navigation, data-entry forms, interactive graph rendering and report previews. JSXsyntax defines reusable components, while CSS Modules or Tailwind CSS handle styling and responsive layouts.

Backend & Pipeline

Python 3.x – primary language for the server: FastAPI endpoints, WhatsApp/Wiki parsers, text normalization, anonymization routines, graph construction (NetworkX), community detection and computation of metrics such as degree, betweenness and closeness. The DWR algorithm and related data-processing tasks are also implemented in Python.

SQL (PostgreSQL dialect) – used indirectly through Xata’s ORM/driver to define and query tables for datasets, user preferences and research metadata.

Optional JavaScript (Node.js) – used for build scripts and tooling (e.g., linting and bundling) but not in the runtime path.

7.5 Risk Management

NetXplore processes conversational data, performs non-trivial computations and interacts with external services. The main risks, their prioritization and mitigation strategies are summarized below.

Risk	Priority	Likelihood /Impact	Mitigation & Controls
Data privacy & anonymization	High	Med / High	Implement pseudonymization by default; allow researchers to opt out; store only hashed identifiers; purge unsaved sessions.
Input quality & schema drift	High	Med / High	Strict parsers for WhatsApp and Wikipedia; schema validation before processing; sample previews; comprehensive parser unit tests.
Scalability on large datasets	High	Med / High	Use streaming and batching during graph construction; cache intermediate results; enable worker concurrency; limit maximum dataset size per session.
External service outages (e.g., Xata)	Medium	Med / Med	Implement retry logic and circuit breakers; provide local fallback storage; queue operations during downtime.
Security vulnerabilities	High	Low / High	Enforce strong authentication for saved projects; follow secure coding practices; sanitize all inputs; limit third-party access permissions.
Cold-start datasets	Medium	Low / Med	Provide default analysis views based on overall network structure; highlight the most connected participants until interactions accumulate.

7.6 Exceptions Management

The system incorporates an exception management mechanism to ensure stability and user friendly error handling.

When unexpected events occur (e.g., invalid file format, missing data, database connection issues, or network errors), the system captures the exception and processes it according to predefined rules.

On the server side, exceptions are logged with detailed technical information for debugging and monitoring purposes, without exposing sensitive details to the end user. The logging mechanism assists developers in identifying recurring problems and improving system reliability.

On the user interface side, error messages are presented in a clear and non technical manner, guiding the researcher toward corrective actions. This ensures that users remain informed without being overwhelmed by system level details.

7.7 Versions Control

Alpha Release – April 3, 2025

Back-end foundations:

Developed the first WhatsApp chat parser to extract participant interactions from .txt files and transform them into structured network graphs.

Implemented fundamental centrality calculations (Degree, Betweenness, Closeness) using NetworkX algorithms to support social network analysis.

Established a FastAPI backend infrastructure with file upload and deletion endpoints for processing chat datasets.

Introduced an initial data validation module capable of parsing WhatsApp formats, including support for multiple date conventions (e.g., US format).

Configured a preliminary database connection with basic CRUD operations for storing research data and user information.

Front-end basics:

Implemented file upload functionality with a drag-and-drop interface for WhatsApp chat files.

Created an initial network visualization using ForceGraph2D, rendering nodes and edges with basic filtering features.

Displayed participant activity metrics and centrality values in a simple tabular format.

Integrated a basic authentication system using JWT tokens and Google OAuth to ensure secure access to the platform.

Beta Release – June 5, 2025

Backend improvements:

Extended data processing capabilities by integrating Wikipedia discussion analysis in .json format, implemented using a Factory Pattern architecture for scalable analyzer design.

Migrated from the initial database setup to PostgreSQL with SQLAlchemy ORM, enhancing data persistence, query efficiency, and research project storage.

Expanded analytical functionality with community detection, comparative analysis modules, and configurable message weight handling.

Developed a comprehensive user management system including profile editing, avatar upload, and research history tracking with persistent storage.

Introduced advanced data filtering mechanisms such as strong-connection filtering based on relative betweenness centrality and adjustable link distance controls.

Deployed the backend with optimized hosting, static file serving, and production-ready configuration for reliability and scalability.

Frontend upgrades:

Completed the research workflow pipeline: platform selection → data upload → filtering → analysis → visualization → export → save.

Enhanced graph visualization with interactive features including node fixation, dynamic scaling, community-based coloring, and customizable layouts.

Implemented a comprehensive filtering system enabling refinement by date ranges, keywords, participants, activity thresholds, and community-based link filtering.

Added robust export capabilities, allowing researchers to download complete datasets and results in Excel and CSV formats.

Built a research project management interface to save, load, compare, and delete analyses, with integrated search and sort functionality.

Designed a comparative analysis dashboard enabling side-by-side discourse network evaluation with specialized metrics and visualization.

Integrated a demo video and user guidance system to support onboarding and assist researchers throughout the analysis workflow.

7.8 Project Management

The development of the NetXplore system was conducted in three structured phases, reflecting standard practices in software engineering lifecycle management. The Core Development phase focused on establishing the system infrastructure, including the setup of a FastAPI-based backend, integration of a PostgreSQL database, implementation of authentication mechanisms (JWT and Google OAuth), and development of a React-based frontend for user interaction.

The Network Analysis phase emphasized algorithmic components, where raw input data from heterogeneous sources (such as WhatsApp chat exports and Wikipedia discussion threads) was normalized, parsed, and transformed into graph-based structures.

During this stage, both classical and advanced network metrics (e.g., Degree, PageRank, Eigenvector Centrality, and Distance Weighted Ranking) were implemented, alongside visualization modules and community detection methods to enhance interpretability. The final Deployment and Reporting phase addressed production readiness, continuous integration and deployment pipelines, performance optimization for large-scale graph rendering, and the generation of exportable reports in CSV, and structured PDF formats. Throughout all phases, project management activities, milestone tracking, and team collaboration were supported via [Notion](#), which provided a centralized environment for task management and workflow transparency.

NetXplore Workflow

NetXplore

▼ Milestones

Task name	Status	Assignee	Priority	Task type
 SRS	Done	T Tslil Aharon נילי S Shira Dagan- Robert:	High	Documents
</> POC	Done	T Tslil Aharon נילי S Shira Dagan- Robert:	High	Code
 SDD Presentation	Done	T Tslil Aharon נילי S Shira Dagan- Robert:	Medium	Presentation
 SDD	Done	T Tslil Aharon נילי S Shira Dagan- Robert:	High	Documents
</> Alpha	Done	T Tslil Aharon נילי S Shira Dagan- Robert:	High	Code
</> Beta	Done	T Tslil Aharon נילי S Shira Dagan- Robert:	High	Code
 Validation Report	Done	T Tslil Aharon נילי S Shira Dagan- Robert:	High	Documents
 Final Presentation	Done	T Tslil Aharon נילי S Shira Dagan- Robert:	High	Presentation
 Project Book	In progress	T Tslil Aharon נילי S Shira Dagan- Robert:	High	Documents
 Defence	Not started	T Tslil Aharon נילי S Shira Dagan- Robert:	High	Presentation

▼ Project Workflow Overview ... +		
Task name	Status	Assignee
✓ Initial Project Setup	Done	ני T Tslil Aharon
✓ Backend API Setup	Done	T Tslil Aharon ני S Shira Dagan- Roberts
✓ Authentication	Done	T Tslil Aharon
✓ File Upload Functionality	Done	ני
✓ Database - PostgreSQL	Done	S Shira Dagan- Roberts
✓ Profile Management	Done	T Tslil Aharon
✓ Spam and Noise Filtering	Done	S Shira Dagan- Roberts
✓ Core Network Analysis	Done	ני T Tslil Aharon
✓ Directed Graph	Done	ני
✓ Network Graph Visualization	Done	ני T Tslil Aharon
✓ Filtering & Highlighting Features	Done	T Tslil Aharon ני
✓ Graph Metrics & Analytics Integration	Done	T Tslil Aharon
✓ Distance Weighted Ranking	Done	S Shira Dagan- Roberts
✓ WhatsApp & Wikipedia Data Parsing	Done	ני
✓ Network Construction	Done	ני T Tslil Aharon
✓ Wikipedia Discussion	Done	ני
✓ Graph Customization Toolbar	Done	T Tslil Aharon
✓ Community Detection	Done	T Tslil Aharon ני
✓ Network Graph Visualization Component	Done	T Tslil Aharon
✓ Dynamic Filtering & Visualization Controls	Done	T Tslil Aharon ני
✓ Comparison Mode & Multi-Network Analysis	Done	T Tslil Aharon
✓ Unified Analysis Architecture	Done	ני
✓ History Page	Done	S Shira Dagan- Roberts
✓ Welcome & Dashboard Page	Done	T Tslil Aharon
✓ Research Creation Wizard	Done	T Tslil Aharon
✓ Research Report Export	Done	S Shira Dagan- Roberts
✓ Performance Optimization for Graph Rendering	Done	T Tslil Aharon
▶ ✓ Data Export (CSV/Excel)	Done	ני
✓ Error Handling & Validation	Done	T Tslil Aharon ני S Shira Dagan- Roberts
✓ Production Deployment Configuration	Done	T Tslil Aharon ני

Figure 7.12 - Development tasks and milestones tracked in Notion

7.9 Code

Git-Hub Repository: <https://github.com/NetXplore-System/NetXplore>

8. System Validation

The validation of the NetXplore system was designed as a structured and multi-layered process aimed at establishing its reliability, robustness, and research value. The evaluation was carried out across three complementary dimensions: functional performance, user experience, and algorithmic correctness. Together, these dimensions provide a holistic assessment of the system's capabilities, ensuring that it not only performs correctly in a technical sense but also offers practical usability and meaningful analytical results for researchers.

To guarantee a realistic and rigorous validation, testing was conducted on heterogeneous datasets drawn from two distinct platforms: WhatsApp group chats and Wikipedia talk pages. These were deliberately selected because of their differing communication dynamics, linear, fast-paced, and informal interactions in WhatsApp versus hierarchical, threaded, and collaborative exchanges in Wikipedia. In total, twenty datasets were analyzed (ten from each platform), covering a spectrum of conversational conditions: routine, low-activity discussions; moderate, incremental shifts in activity; and high-intensity crisis events that drive sudden bursts of communication. This diversity provided a robust basis for examining the system's behavior under varying engagement levels.

To guide and structure the evaluation, we defined four core validation criteria:

- **Accuracy** – Network structures, interaction graphs, and computed metrics must faithfully represent the original discourse, without distortion or information loss.
- **Responsiveness** – The system should maintain acceptable performance across different dataset sizes, from small WhatsApp groups with a dozen participants to extensive Wikipedia debates with hundreds of contributors.
- **Clarity of Visualization** – Graphical outputs should remain interpretable and visually coherent even for dense, high-volume datasets, highlighting meaningful interaction patterns and community structures.
- **Cross-Platform Consistency** – Analytical outputs should be comparable across different platforms; the same concepts (e.g., participant centrality, community detection) should be consistently detected whether the data originates from WhatsApp or Wikipedia.

The validation strategy combined quantitative testing and qualitative evaluation. On the one hand, controlled experiments and systematic test cases ensured that NetXplore's computational modules from parsers and graph constructors to metric calculators—behaved as expected. On the other hand, usability testing and expert reviews provided insight into how researchers experience the platform in practice. Importantly, an independent cohort of network analysis experts, coordinated by

Mr. Assaf Shapira, tested the system with representative datasets and provided structured feedback. Their input was vital in confirming not only the correctness of the outputs but also the accessibility, clarity, and overall research value of the tool.

The validation results consistently confirmed that NetXplore is:

- Functionally robust, reliably ingesting, processing, and visualizing discourse data end-to-end.
- User-friendly, lowering the barrier of entry for non-technical researchers while offering advanced functionality for experts.
- Algorithmically correct, producing outputs that align with both theoretical expectations and real-world communication dynamics.
- Cross-platform consistent, generalizing analytical methods across heterogeneous sources without compromising interpretability.

The following subsections detail the validation outcomes for each of these dimensions: Functional Validation, Usability Testing and Expert Feedback, Algorithmic Accuracy and Analysis Correctness, and System Robustness and Cross-Platform Consistency providing a comprehensive view of how NetXplore was tested and how it performed across scenarios.

8.1 Functional Validation

Validation Strategy

Three representative categories of conversational scenarios were defined to guide the evaluation:

- Stable periods (“nothing happens”) – sessions characterized by low or routine interaction levels.
- Minor events – small disturbances, such as an influx of new participants or minor topical shifts.
- Major events – intense, real-world crises that trigger sudden spikes in communication volume and structural reorganization of discourse.

By testing across these categories, we verified whether NetXplore adapts its analytical outputs to reflect the true dynamics of the conversation. Four key validation criteria were assessed throughout: accuracy of network structures, responsiveness under varying loads, clarity of visualization, and cross-platform consistency.

High-Activity Scenarios (Major Events) During high-intensity events, NetXplore correctly captured both the volume of activity and the structural changes within discourse networks.

Case 1: Wartime WhatsApp Group.

On the day of a major real-world conflict, activity within a WhatsApp group surged to 72 active users generating 331 directed interactions. NetXplore reported a significant rise in density (≈ 0.065) for a network of this size, reflecting an unusually connected discourse, and a reciprocity of ≈ 0.23 , highlighting frequent back-and-forth communication. Visualization outputs revealed a tightly connected network, while community detection identified four distinct clusters that matched the actual emergent themes of discussion, emotional support, live updates, coordination of resources, and opinion exchange.

Case 2: Flash Flood in Thailand.

In a travel-focused WhatsApp group, casual conversation abruptly shifted into urgent coordination during a natural disaster. NetXplore captured this transition: 62 participants engaged in 87 interactions over a short period, with reciprocity increasing to ≈ 0.08 . The visualization showed a dense network core of highly active users providing critical updates and guidance. Community detection identified eight task-oriented clusters, including evacuation planning, weather monitoring, embassy contacts, and accommodation support. This outcome aligned perfectly with the expected structure of group communication under crisis pressure.

These major-event tests validated NetXplore's ability to dynamically reflect spikes in connectivity, identify central figures, and reveal topical subgrouping all hallmarks of crisis-driven conversations. Moderate and Low-Activity Scenarios In moderate or low-activity cases, the system demonstrated sensitivity to subtle variations in discourse without over-interpreting noise.

Case 3: Routine Questions (Minor Event).

When new members joined a WhatsApp group and asked repetitive questions, NetXplore logged a moderate rise in activity (51 users, 101 interactions). Network density (~ 0.04) and reciprocity (~ 0.15) reflected balanced, though not extreme, engagement. Centrality measures highlighted long-term members who repeatedly answered questions, and community detection identified six thematic clusters corresponding to recurring topics such as visas, travel tips, and cultural advice.

Case 4: Quiet Day (Nothing Happens).

On a routine day in the same group, only 23 users exchanged 34 interactions. NetXplore generated a sparse graph (density ≈ 0.067 , reciprocity ≈ 0.18), with no dominant clusters or hubs. This outcome validated that the system correctly portrays stable, decentralized structures when no significant events occur, providing a baseline reference against which to contrast high-activity periods.

Cross-Platform Validation (Wikipedia Talk Pages) NetXplore's consistency was further confirmed by applying it to Wikipedia talk pages, which differ from WhatsApp in being threaded, asynchronous, and editorially focused. Despite this structural difference, the system produced coherent and accurate outputs.

Case 5: Political Protest Discussion.

A thread of 33 comments by 12 contributors was processed into 11 nodes and 26 interactions. Metrics indicated moderate connectivity (density ≈ 0.19 , reciprocity ≈ 0.19 , diameter = 5). Community detection found four clusters corresponding to distinct editorial stances.

Case 6: Contentious Edit Debate.

In another talk page case, NetXplore revealed two polarized clusters, each aligned with one side of a content dispute. The density (~ 0.15) reflected limited dialogue between the factions, accurately mirroring the discourse structure of Wikipedia debates.

These results confirm that the system generalizes well across platforms: metrics were interpretable in both contexts, reciprocity values held comparable meaning, and centrality scores consistently highlighted influential users.

Performance and Responsiveness Throughout all functional tests, NetXplore maintained robust performance. Networks with only a few dozen nodes and others with 70+ active participants were processed without lag or system errors. Even in heavy-load scenarios with hundreds of messages, graph construction, metric computation, and visualization updates were completed smoothly. This confirmed compliance with the responsiveness criterion, ensuring scalability across dataset sizes.

8.2 Usability Validation

The usability validation assessed whether NetXplore is not only technically robust but also practical and intuitive for research workflows. To ensure independence, the evaluation was conducted by a social network analysis team led by Dr. Asaf Shapira, who tested the system with representative datasets in a full end-to-end workflow.

Expert Review Process

The experts engaged with the system autonomously, covering all stages: data import, preprocessing, network construction, metric computation, visualization, and report export. This hands-on approach ensured that feedback reflected natural usage rather than guided demonstrations.

Findings

Interface and Learning Curve

The interface was described as clean, logically organized, and easy to learn. Interactive tooltips and immediate feedback (e.g., graphs updating dynamically upon filter changes) enabled even first-time users to perform complex analyses with minimal effort.

Visualization and Interpretability

Graph visualizations were highlighted as a major strength. Nodes, edges, and community clusters were presented clearly, while node sizing and coloring by centrality measures effectively emphasized key actors. Hovering tooltips provided detailed metrics without cluttering the view. The experts confirmed that the platform met the clarity criterion, delivering interpretable results even in dense networks.

Workflow Integration

The seamless combination of graphs, metric tables, and timeline charts supported fluid transitions between macro-level exploration (e.g., detecting spikes in activity) and micro-level inspection (e.g., identifying a specific user within a cluster). This integration removed the need for external tools and enabled comprehensive analysis in one environment.

Report Generation

The PDF reporting feature was praised for producing well-structured and publication-ready documents. Reports included network visualizations, statistical summaries (e.g., centrality, reciprocity), and narrative highlights, making them valuable for both technical and non-technical audiences.

Practical Research Value

For non-technical users, NetXplore was seen as lowering barriers to advanced network analysis by eliminating the need for programming skills. For experienced analysts, it was recognized as a time-efficient exploratory tool, accelerating the discovery of interaction patterns and community structures. This allowed researchers to focus more on interpretation and theory-building rather than technical setup.

Areas for Improvement

While feedback was overwhelmingly positive, experts suggested minor refinements:

Improved label placement for very dense graphs.

More granular filtering options (e.g., by message type or interaction direction) to support large or multifaceted discussions.

These suggestions were integrated into the iterative development process, leading to refinements in the visualization engine and filtering interface.

8.3 Algorithmic Validation

A critical component of the validation process focused on ensuring that NetXplore's analytical algorithms are both correctly implemented and yield results that are meaningful in real-world discourse analysis. Since the system's insights rely heavily on the robustness of its computational methods, we performed targeted tests of three core algorithmic pillars: the Distance Weighted Ranking (DWR) mechanism for reconstructing conversational threads, the community detection module for clustering users, and the suite of centrality measures for identifying key participants.

Distance Weighted Ranking (DWR)

The custom DWR algorithm was designed to weight interactions according to their temporal proximity in the conversation, thereby modeling the natural recency effect in human discourse. To validate its implementation, we applied it to a controlled WhatsApp dataset (the "Gafni Team" group, November 25–29, 2024). In this configuration, each message was linked to up to three preceding messages, with weights of 0.5, 0.3, and 0.2 respectively. Manual tracing of the interaction timeline confirmed that NetXplore's generated edges precisely matched the theoretical weighting scheme: every participant-to-participant link accumulated weights according to the correct coefficients. A weight table exported from the system validated that no anomalies occurred, and that the decay structure (heavier emphasis on immediate replies, lighter emphasis on earlier ones) was faithfully captured.

This validation demonstrates that DWR is not only technically correct but also conceptually valid. By embedding conversational context into the network graph, the algorithm ensures that recent

and direct interactions carry more influence in centrality and community calculations, thereby reflecting real patterns of conversational dynamics.

Community Detection

The community detection module was validated through case studies that compared computed clusters to thematic substructures known to exist in the data. For example:

In the wartime WhatsApp group on October 7, 2023, NetXplore detected four distinct communities. Manual review showed that these corresponded to actual conversational threads such as emotional support, coordination of actions, and news-sharing.

In the Thailand flood scenario, the system detected eight sub-communities, aligning with task-oriented clusters (e.g., evacuation planning, weather updates, embassy contact, emotional support).

Validation also confirmed that the algorithm’s sensitivity scales with discourse fragmentation. When the war-related chat later diversified into six parallel subtopics, the number of detected clusters increased accordingly. Conversely, during quieter “no-event” scenarios, the tool identified only loosely knit, minor clusters, consistent with expectations for diffuse conversations.

These results affirm that the community detection module faithfully partitions networks in line with actual social structures, ensuring researchers can reliably interpret groupings as meaningful sub-communities.

Centrality and Influence Metrics

NetXplore implements a standard set of centrality measures – Degree, Betweenness, Closeness, Eigenvector, and PageRank – to assess participant influence. Validation focused on both technical

correctness and contextual validity: Correctness: Simple verification showed that top-degree users indeed had the highest number of edges in the raw data, and PageRank values reflected proper normalization consistent with link structure.

Contextual validity: Across scenarios, the system reliably highlighted participants who played pivotal roles in the conversations. For example:

In the flood emergency group, the highest-centrality users were the de facto coordinators who disseminated information and answered questions.

In the “new members” scenario, veteran participants were identified as central hubs by repeatedly responding to newcomers.

A high degree of convergence across centrality measures was observed: the same top five users typically ranked highest in multiple metrics, increasing confidence that the algorithms were correctly implemented.

One anomaly was noted in a Wikipedia talk page case, where two participants were initially excluded because they signed comments in a non-standard way. This was traced to a parsing issue rather than an algorithmic fault. Once corrected, centrality and community outputs updated logically, further confirming the algorithms’ correctness given clean input.

Summary of Algorithmic Validation

The algorithmic validation provides strong evidence that NetXplore’s analytical engine is both technically sound and empirically reliable. The DWR algorithm was proven to correctly apply recency-based weights, enhancing the realism of conversational reconstruction. Community detection consistently revealed clusters with clear thematic or functional interpretations,

reflecting actual discourse substructures. Centrality metrics were accurate in both computation and interpretation, reliably flagging the true influencers and hubs within conversations.

Overall, these validations demonstrate that NetXplore's algorithms not only function correctly at a computational level but also generate socially interpretable and analytically meaningful results. This gives researchers high confidence that the platform can be trusted for rigorous analysis of online conversational networks.

[Link to our validation document](#)

9. SUMMARY, EVALUATION, CONCLUSIONS AND FUTURE WORK

9.1 Summary

NetXplore was developed as a dedicated platform for network researchers, designed to transform digital conversations from WhatsApp and Wikipedia into structured interaction networks. By providing tools for visualization, filtering, anonymization, and algorithmic analysis, the system enables researchers to explore communication patterns, identify central participants, and study the structure of discussions.

The platform emphasizes flexibility, giving researchers the ability to adjust analysis parameters and choose how data is represented. It also supports the creation of research reports that summarize findings and allow for further academic study.

In essence, NetXplore provides a bridge between unstructured digital conversations and meaningful network insights, offering a clear and scalable approach to understanding human interaction in online communication spaces.

9.2 Evaluation and Conclusions

NetXplore successfully demonstrates the ability to transform unstructured digital conversations into structured network representations. The system provides researchers with practical tools for filtering, analyzing, and visualizing interactions in a way that is both intuitive and academically valuable.

The evaluation of the system shows that it meets its main goals:

- Supporting researchers in analyzing conversations from WhatsApp and Wikipedia.
- Offering flexible filtering and weighting mechanisms that adapt to different research needs.
- Providing a clear and user-friendly UI that makes advanced network analysis accessible even for users with limited technical backgrounds.
- Ensuring scalability for future extensions to additional platforms.

9.3 Future Work

Looking ahead, NetXplore has the potential to evolve in several meaningful directions. While the current version supports WhatsApp and Wikipedia discussions, the system was intentionally designed in a scalable way that will allow integration with additional platforms in the future.

Another important direction is expanding the algorithmic capabilities, including more advanced weighting schemes, clustering methods, and possibly even optional content-based analysis for researchers who require it.

On the user experience side, future work may focus on improving the UI with more interactive features, richer visualization options, and enhanced customization to better fit diverse research needs.

Finally, as the system grows, considerations around performance optimization and collaborative research features will become increasingly important, enabling multiple researchers to share, compare, and co-develop studies within the platform.

On the visualization side, NetXplore could incorporate more advanced exploration features. Tools like Connected Papers demonstrate how knowledge networks can be navigated intuitively through interactive graph representations (Connected Papers, 2020)[5]. Inspired by such platforms, NetXplore could extend its visualization layer to enable researchers to explore conversations dynamically, compare multiple datasets in a single view, and highlight thematic or structural overlaps.

Another promising direction is the integration of Natural Language Processing (NLP) techniques. While the current version of NetXplore focuses solely on the structural aspects of conversations, future extensions could incorporate semantic analysis to detect sentiment, identify key topics, and trace how participants' positions evolve throughout a discussion. This combination of structural and semantic insights would provide researchers with a more comprehensive understanding of online interactions (Jelodar et al., 2017)[6].

10. REFERENCES

- [1] Chang, J. P., Chiam, C., Fu, L., Wang, A. Z., Zhang, J., & Danescu-Niculescu-Mizil, C. (2020). ConvoKit: A Toolkit for the Analysis of Conversations. *Proceedings of the 21st Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 57–60. <https://aclanthology.org/2020.sigdial-1.8>
- [2] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3, 993–1022. <https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf>
- [3] Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks. *Proceedings of the International AAAI Conference on Web and Social Media*, 3(1), 361–362. <https://gephi.org>
- [4] Choudhury, T., & Basu, S. (2004). Modeling conversational dynamics as a mixed-memory Markov process. *Advances in Neural Information Processing Systems (NeurIPS)*. Retrieved from <https://papers.nips.cc/paper/2624-modeling-conversational-dynamics-as-a-mixed-memory-markov-process>
- [5] Connected Papers. (2020). Connected Papers: A visual tool to explore academic papers and their connections. Retrieved from <https://www.connectedpapers.com/>
- [6] Jelodard, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., & Zhao, L. (2017). Latent Dirichlet Allocation (LDA) and Topic Modeling: Models, applications, a survey. *arXiv preprint*. <https://arxiv.org/abs/1711.04305>

NetXplore

הנדסת תוכנה ספר פרויקט גמר לתואר ראשון בוגר במדעים Sc.B

מאת:

צליל אהרון, נוי רייכמן, שירה דגן-רוברטס

מנחה:

דר' דביר רוס

מוגש כחלק מהדרישות לקבלת תואר ראשון בוגר במדעים (B.Sc)

תקציר

NetXplore היא פלטפורמת אנליזה מתקדמת, המיועדת לעיבוד, השוואה והמחשה של שיח חברתי ממקורות הטרוגניים, ובראשם שיחות WhatsApp ודפי שיחה בוויקיפדיה. המערכת מיישמת תהליך עבודה מקצה לקצה החל מקליטת הנתונים ואנונימיזציה, דרך בניית גרף וניתוח אינטראקטיבי, ועד לשילוב אלגוריתמי זיהוי קהילות, מדדי רשת, ושיטת דירוג חדשנית בשם *Distance Weighted Ranking*, על מנת לחשוף תובנות מבניות והתנהגותיות החבויות בטקסט הגולמי.

מאפיין מרכזי של NetXplore הוא המחשה גרפית אינטראקטיבית, שבה המשתתפים מיוצגים כצמתים והאינטראקציות כקשתות. כך יכולים חוקרים לחקור באופן דינמי את הקשרים, להבליט דמויות מפתח, ולאתר מבני קהילה ותבניות תקשורת שלא ניתן לזהות בקריאה ידנית.

מעבר להמחשה, הפלטפורמה מציעה כלים לסינון והשוואה, המאפשרים לנתח שיח על בסיס טווחי זמן, מילות מפתח ורמות מעורבות, וכן לבצע השוואות בין מערכי נתונים זה לצד זה. על ידי גישור בין יומני WhatsApp הליניאריים לבין שרשרים היררכיים בוויקיפדיה, NetXplore מספקת ממשק אחיד, ידידותי לחוקר, שתומך גם בחקירה איכותנית וגם במדידה כמותית ובסופו של דבר הופכת שיחות גולמיות לתובנות משמעותיות המבוססות על המחשה חזותית.

הכרת תודה

ברצוננו להביע את תודתנו הכנה לד"ר דביר רוס על ההכוונה, העידוד והתובנות המקצועיות שהיו בעלות חשיבות מכרעת בעיצוב כיוונו של פרויקט זה. מסירותו, זמינותו והמשוב הבונה שהעניק לנו ליוו אותנו בכל שלב והעשירו מאוד את התהליך והתוצאה.

אנו מבקשים להודות לפרופ' אביבית לוי, ראש החוג להנדסת תוכנה, על ההדרכה, ההכוונה והתמיכה המתמשכת לאורך הדרך, וכן על החיבור לאסף שפירא אשר תרם משמעותית להצלחת הפרויקט.

כמו כן, אנו מודים לאסף שפירא על תרומתו לוולידציה של המערכת, על חיבורנו לקבוצת חוקרי רשתות, ועל ההערות והתובנות החשובות שהעביר לנו מתוך ניסיונו המחקרי, אשר תרמו לשיפור המערכת והעניקו לנו נקודת מבט רחבה מהתחום.

לבסוף, אנו מבקשים להודות לפקולטה להנדסה בשנקר על המסגרת האקדמית והמשאבים שסיפקה, אשר אפשרו את פיתוחו המוצלח של פרויקט NetXplore.