Bilkent University
Department of Computer Engineering

Senior Design Project

T2334

PaperAtlas

Final Report

Ahmet Hakan Yılmaz - 21803399

Akın Kutlu - 21803504

Aybala Karakaya - 21801630

Selbi Ereshova - 21901326

Zehra Erdem - 21801977

Supervisor: Uğur Doğrusöz

Jury Members: Erhan Dolak and Tağmaç Topal

19.05.2023

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

1. Introduction

Paper Atlas is a web application available on all modern web browsers. Paper Atlas shows the papers as nodes with edges specifying which one references which one and which one is being referenced by which. In this way, the project enhances the way academicians and students conduct their research by providing a better way to find papers that best match their research topic by showing the relation of papers. The user is able to read the details such as name,study areas, and abstract by clicking on the node of a paper. A link to the original paper is available as well. The graph is interactive; in other words, the users can move the nodes around, zoom in and out, apply layouts, and highlight nodes. Users can filter their results by time interval, study area and citation amount. Users can also search authors instead of paper titles. In this case, relative nodes will be shown for authors. The sizes of nodes within the graphs change according to their influence.

2. Requirements Details

2.1. Functional Requirements

- Viewing papers, authors and their relations as an interactive graph.
- Applying different kinds of layout to the graph.
- Searching papers by their titles and viewing the results.
- Searching the authors by their names and viewing the results.
- Creating a new graph from selected search results.
- Merging selected search results to the current graph.
- Bringing references or citations of the selected papers within a chosen distance after search.
- Bringing papers of a selected author to the graph.
- Bringing authors of a selected paper to the graph.
- Bringing references of a selected paper to the graph.
- Bringing citations of a selected paper to the graph.
- Finding common references of more than one selected papers.
- Finding common citations of more than one selected papers.
- Finding common papers of more than one selected authors.

- Removing nodes from the graph.
- Applying time filter to the papers in the graph
- Applying field areas filter with ”and” and “or” options to papers in the graph.
- Applying citation count filter to the papers in the graph.
- Pinning nodes so that they will not be eliminated by any filter.
- Downloading the current graph.
- Uploading a graph.

2.2 Non Functional Requirements

2.2.1 Performance

The response time for the application besides running layout and retrieving a graph from the database should not be more than 30 milliseconds. But since a graph can be very large, running layouts and retrieving such graphs from the database can take longer but should not exceed 30 seconds.

2.2.2 Usability

The user interface of the application should be user friendly. Everyone using the application should be able to understand how to handle functionalities of the application. As the application is based on graph visualization, graphs should have interactive design for everyone to use it easily.

2.2.3 Maintainability

To be able to maintain the application, the code should be easy to read and map with requirements so that when changes are required it will be easy to locate them. The code should be written in a way to separate different tasks into different modules so that functions can be modified without affecting each other too much and new functions can be added easily to maintain an application over time.

2.2.4 Scalability

The application must be able to handle 10000 users at the same time. Moreover, as the initial goal of the application is to provide services for specific

topics, the application should be able to continue to function if it is decided to provide services for other topics.

2.3 Pseudo Requirements

2.3.1 Implementation Constraints

Node.js framework is used for the implementation of backend with Javascript. Cytoscape.js library is used for visualization of the graphs. Neo4j which is a graph database is used to store information. Cypher is used for graph query language. Response time is less than 30 seconds.

2.3.2 Data Constraints

Publicly published papers are used as data. Publicly accessible information of papers such as author, title and abstract is stored and used.

2.3.3 Cost Constraints

To gather information about papers and authors, free APIs are used.

2.3..4 License Constraints

For all the used libraries, APIs and frameworks, The ones that their licenses are appropriate for the project are used.

2.3..5 Schedule Constraints

Before the first semester ended, a prototype demo was done. Before 19 May 2023, the project was done.

3. Final Architecture and Design Details

3.1 Overview

In this section, the final architectural structure and design details of Paper Atlas will be explained. Subsystems and their purposes and subsystem components and their purposes will be explained. Paper Atlas follows a client-server architectural

pattern. In the client side there is Frontend Layer and in the backend layer there are Backend layer, Database layer and Third Party API service. Most of the required information for the Paper Atlas is stored in our Neo4j database. Client side sends REST requests with required inputs from user to server. The connection between server and database is provided with ODBC. Both the server and client sides are deployed on render.com.

3.2 Subsystem Decomposition

Our system follows the client server architecture. In our client side we have the frontend layer and in the server side we have the backend layer, one database layer for the Feedback Database, one database layer for the Paper Author Database and Third Party API. The whole system is composed of four main parts: Frontend layer, Backend layer, Database layers and Third Party API.

In the Frontend layer we have UI components that the user will interact with in order to use our application. Canvas UI is used to display the nodes and relationships representation of the queries and searches the user makes. To bring data to the Frontend layer the users can use Search UI and send a request to the Backend layer. Users will be able to obtain the results of each node via Node detail UI. They will also be able to filter the nodes and relationships on the Canvas UI via interacting with Filter UI. The users are allowed to make changes to data in Canvas UI while interacting with the Query UI as well.

In the Backend layer which is the connection between frontend and database layer and third party api we have management components. The request management is responsible for requests coming from frontend. Node and relation management prepares the result for query in wanted format and when database or api is needed, it communicates with necessary service. Database services have access to database layers and third party API service can get data from third party api.

We have two database layers.

One database layer contains only one component which is the Paper Author Database. It is a graph type of database and a Neo4j database. All data related to this project is in this database. Backend has a connection to the database.

The other database layer contains only one component which is the Feedback Database. It is a Postgresql database. All data related to the feedback collected from the users on our website is in this database. Backend has a connection to the database.

The last part is a third party api. The backend has a direct connection to the Semantic scholar api. This connection is used to get more detailed information for papers such as abstract and access url.

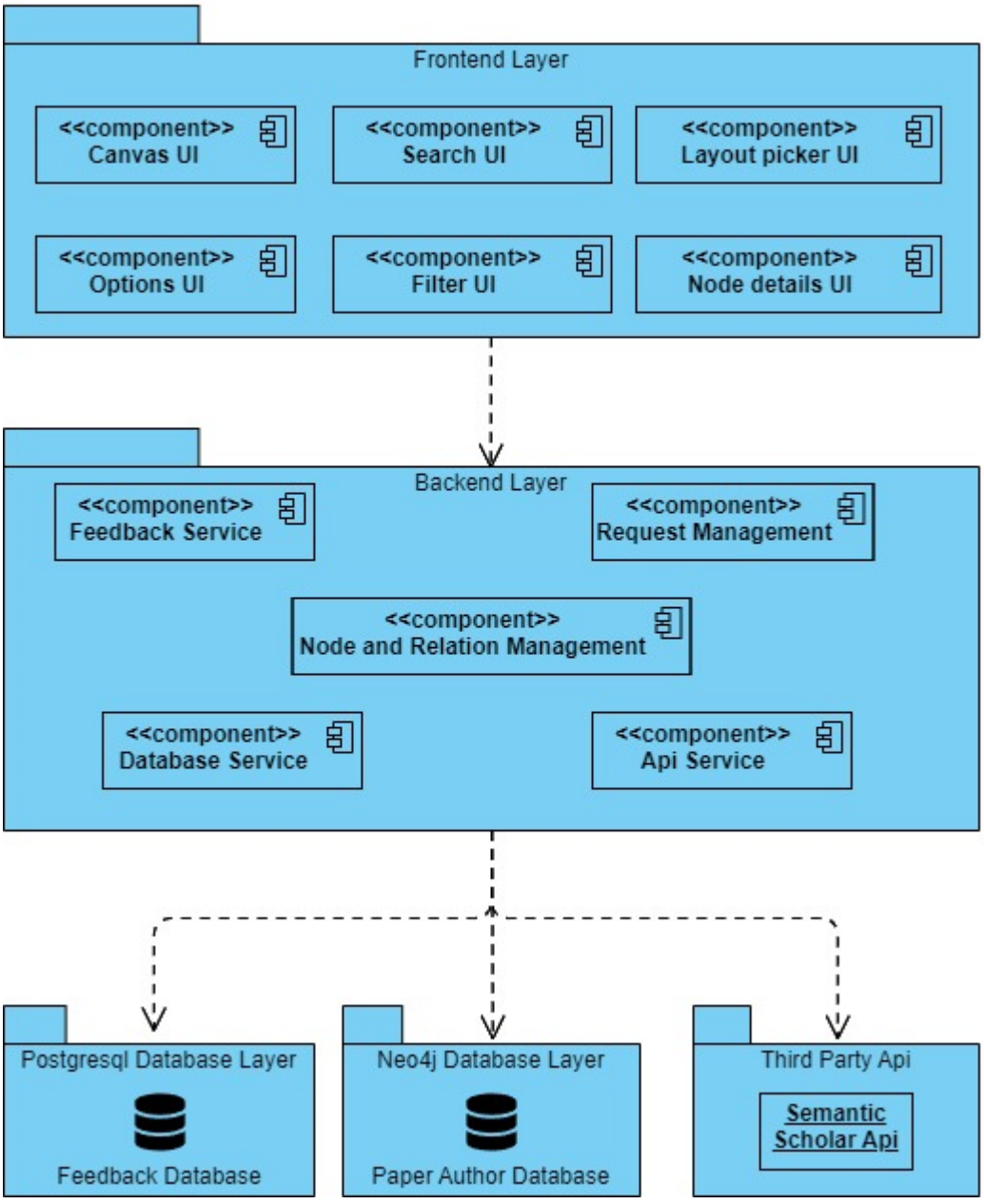

Figure 3.2: Subsystem decomposition diagram

3.3 Hardware/Software Mapping

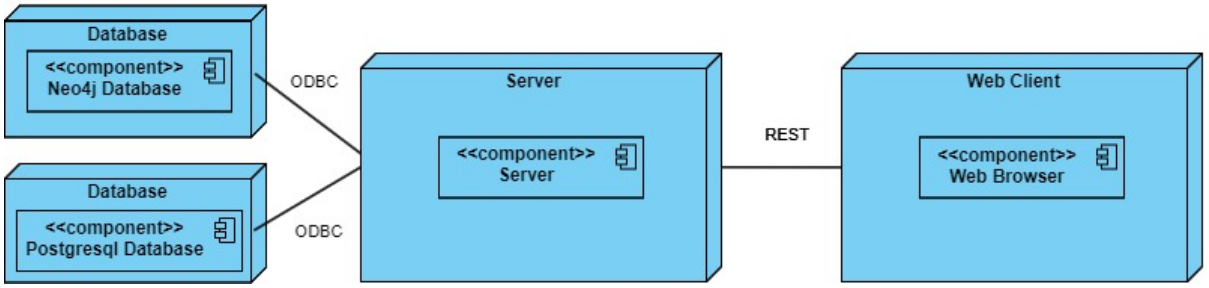

Figure 3.3: Hardware/Software Mapping

The paper and author information is stored in a Neo4j database. The server performs Cypher queries on the database by connecting to it using ODBC. The functions to create graphs are implemented on the server side. The client sends REST requests to the server with the input it gets from the users to create the graphs in the requested format.

The feedback information which is collected from the users on our website is stored in a Postgresql database. The server performs SQL queries on the database by connecting to it using ODBC. The functions to send and get feedback are implemented on the server side. The client sends REST requests to the server with the input it gets from the users to perform the requested operation.

The server and the client are deployed on render.com separately. The client can be run on any web browser.

3.4 Persistent Data Management

The relations between nodes are important so our project requests a special type of database: Graph Database. We decided to use Neo4j because it is open source and it meets our needs. The data in our database downloaded from semantic scholar. Before pushing the data, it was filtered. There are 2 types of nodes in our database: Author and Paper. There are 2 types of relations: a-reference-of and an-author-of.

Number of nodes and relations in the Graph database

Author nodes: 339857	Paper nodes: 71900	Total nodes: 411757
a-reference-of relations: 75155	an-author-of relations: 449376	Total relations: 524531

The schemes of nodes and relations

Paper

citationCount

venue

journalName

uniqueFieldsOfStudies

year

publicationTypes

acl

title

dblp

journalPages

url

mag

pubmed

referenceCount

arXiv

influentialCitaitonCount

journalVolume

isOpenAccess

pubMedCentral

publicationDate

paperId

doi

Author

citationCount

aliases

paperCount

orcids

name

affiliations

hindex

authorId

url

dblps

homepage

an-author-of

identity

start (authorId)

end (paperId)

type

a-reference-of

identity

start (paperId)

end (paperId)

type

We also have another database to store the feedback gotten from our users: Feedback Database. We decided to use Postgresql because it was enough for us to be able to perform SQL queries. There are 1 type of data in this database: Feedback. Its fields are given below.

Feedback

id

name

surname

point

message

mail

3.5 Boundary Conditions

Paper Atlas has three types of boundary conditions which are initialization, termination, failure.

Initialization:

Users should have an internet connection to use Paper Atlas. Paper Atlas can be used from any device which has an Internet connection. Since Paper Atlas does not have login and registration cases any user with internet connection can use PaperAtlas. For the best experience a computer should be used as its screen is the most suitable for graph visualization.

Termination:

Closing the web page or going to another web page terminates Paper Atlas.

Failure:

If Internet connection is cut while in using the Paper Atlas network failure can occur. There may be a need for some updates in the database as time passes.

During the updates, the database server can be down and data cannot be reached by the server which would cause a failure.

3.6 Subsystem Services

Request Management

Within this module, the http requests from the frontend are received. The necessary functions of the Node Management module are called. When the response data is created by the Node Management module, it gets formatted and sent as an http response.

Node and Relation Management

This module is where business logic happens. Within this module, functions specific to requirements are implemented. This module calls the Database Service or External API Service according to need and get wanted data. After that, business logic is applied and the resulting data is processed into desired format.

Database Service

This module communicates with Database Servers. Within this module queries that return desired data are written and run on the Neo4j Database Server and Postgresql Server.

API Service

This module communicates with any third party APIs. For the time being, Scholar API is needed to retrieve some data. Therefore, this module is responsible for retrieving data from the Scholar API.

Canvas

Canvas is the component where the graph is displayed and nodes are displayed visually. From the nodes on the graph new nodes and edges can be merged. In this situation new requests are sent to the backend layer. If wanted, nodes and edges can be deleted from the canvas. Elements of the graph can be moved within the canvas as well.

Search

Search component is the component where users can search for authors and papers. As a result of searching new requests are sent to the backend layer to bring the new information for the canvas.

Layout Picker

Layout Picker component enables users to select the layout of the graph from a list of predefined graph layouts. When a layout is chosen, it will be applied to the graph in Canvas.

Options

Options component is the component that enables users to do different tasks such as finding common ones or uploading and downloading of the graph.

Filter

Filter component enables users to apply different filters to the graph in the Canvas.The filtering is done in the frontend layer. Some examples of available filters are filter by time,citation amount, and by topics.

Node details

When a node in the canvas is clicked, the Node Details component displays the details of a node. Details of nodes are brought from our database and the semantic scholar api via backend layer as in our database all details of nodes are not stored.

3.7 Class Diagram

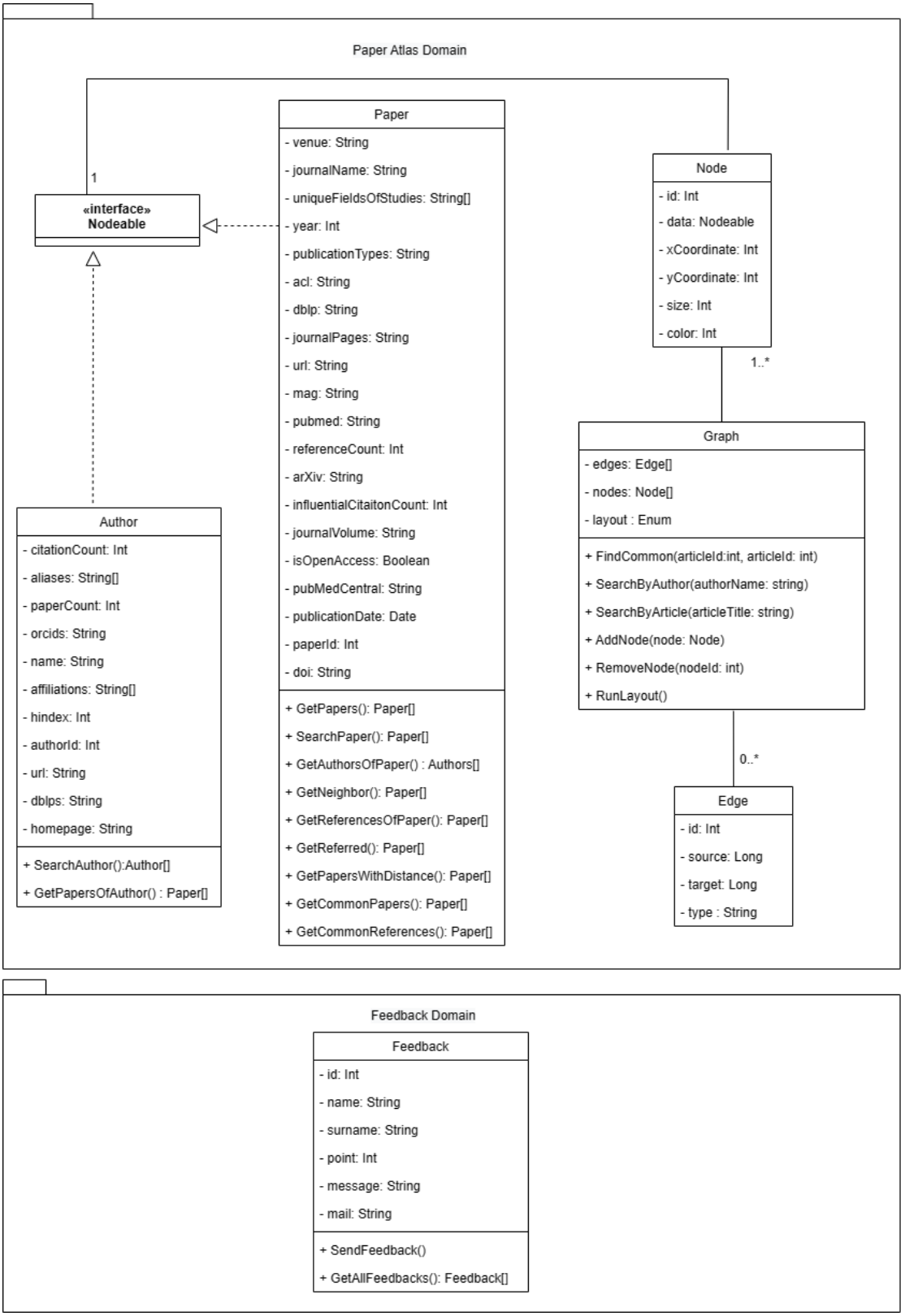

Figure 3.7: Class Diagram

Graph

The graph class is the main part of the project and it is for the representation of the whole graph. It basically consists of edge and node objects. In order to control and change the graph, layout property is used.

Node

Node is a representative class for nodes in a graph. It has a nodeable instance to keep data. It also has coordinates to keep the position of the node in the graph. It also has color and size attributes which can be set according to the information in the nodeable attribute.

Edge

Edge is a representative class for directed edges in a graph. Therefore, it hase source and target attributes. It also has type attributes since it is required to use different types of edges between different kinds of nodes.

Nodeable

Nodeable class is an interface which is inherited by Author and Article classes. Those classes are the classes which can be represented as a node in Paper Atlas.

Author

Author class is for the representation of the authors of articles in our database. It has citationCount, aliases, paperCount, orchids, name, affiliations, hindex, authorId, url, dblps and homepage properties. It has SearchAuthor() and GetPapersOfAuthor() functions.

Paper

The purpose of this class is to create a respective class Paper for Paper nodes in our graph database. This class has the same properties and variables as the Paper node in the database. This class also implements relevant methods for the Paper node such as getting authors of the article, finding references of the article or finding referred papers or both, searching for an article, finding papers within a given

distance in terms of referring them or being referred by them or both, finding papers commonly referred or referred by.

Feedback

The purpose of this class is to be able to store the feedback taken from the users in the Postgresql database so that we can access them easily. “id” property is the id of the feedback. “name” and “surname” properties store the name and the surname of the user who created the feedback. “point” property stores how many points are given to the feedback. “message” property stores the feedback message. “email” property is the email of the user who created the feedback.

4. Development/Implementation Details

4.1 Backend

In the backend we use Node.js along with axios.js library.

We have a structure to divide the logic of our backend between the endpoints, database controllers, Cypher queries and data models.

server.js - in this file we have all our endpoints, calling respective controller functions.

database-controllers.js - in this file we have all our controller functions called by the endpoints. The controller calls functions from basic-queries.js to get respective Cypher queries. The database controller runs the queries using the axios library. The results of these queries are processed in controllers as well before being back passed to endpoints.

basic-queries.js - in this file we are creating all our Cypher queries necessary for retrieving data from our Neo4j database. Each controller in the database-controller calls a respective function in this file to return a query in string format. We decided to have these queries in a separate file to reduce the complexity of the files and since the queries can be really long in length.

models.js - in this file we have the data model for the Feedback object.

.env - in this file we store all the necessary credentials to connect to our local, cloud databases.

By separating the code into many different files we ensured the maintainability of our code and made sure we followed standards by many developers and organizations.

4.2 Frontend

In the frontend we use React as it is easy to implement and can support many other technologies we may implement. We tried to break down the whole page to smaller components to make implementation easier. We have typescript files for each of the components we create in the user interface. In React you can update a constant value by using useState hook. Each component was able to preserve the state of their own variables, however, we needed to update our graph elements from different components. Because of that we stored our graph elements in the main component that renders other small components such as Node Detail, Search and Filter. Whenever an update was needed for the graph from a rendered component, we wrote the updating functions to the main component and sent it down to the needed components by using props. Therefore, we were able to manage elements from one place and were able to preserve their states.

We also use “react-cytoscapejs” library to be able to use Cytoscape. Cytoscape is to visualize a graph with styles. The library makes it able to render a Cytoscape Component. We used this component and its attributes such as stylesheet to show our graph. We used Material UI for the components such as buttons, a slider, select, drawer or input labels to give them more style and to have more user friendly interfaces.

4.3 Database

Neo4j (Paper Author DB):

In the first implementation of the project, an online neo4j database was chosen because we wanted to create only one database and all team members had access to the same database. There are lots of limitations of this online neo4j database: maximum number of nodes and relation. Also, uploading was hard for bulk data. In the next stages of the project, we created another neo4j database. This one is a local database and we expanded our database approximately 3 times easily. We

created copies of our database and contents to back up. This version allowed us to create a database much bigger in a shorter time. For example, in the first version, there are only 2 types of papers in the database and the new version has all 23 types of paper. For each version, the data is downloaded from Semantic Scholar and formatted in order to upload our database successfully.

Postgresql (Feedback DB)

In the first design of the project, there was no Postgresql database. However, in order to collect user feedback, we needed a SQL database. We created an online Postgresql database on supabase.com. The only purpose of this database is feedback.

5. Test Cases and Results

Every test case has a unique ID and name. What is the purpose of the test case is explained in the Summary. Post procedure and procedure of the test cases and expected results are written. Every test case has a category and security result. There are two main categories which are Functional and Nonfunctional. Nonfunctional ones have different sub categories which are Performance to test mainly time performance of the application, Reliability to test correctness of the application and Usability to test whether necessary messages are given to the user or the thing makes it easier to use application. There are three kinds of severity which are critical to emphasize these cases are very crucial for the application to work, major to emphasize important cases and minor to emphasize that they do not have several effects on the application.

Table 5: Test Cases

Test IDs	Test case name	Summary	Post Procedure	Procedure	Expected Result	Category	Severity Result	Test results
T01	Searching papers	Testing if we get list of papers when searching for papers	-	Search different keywords when search type is Paper	There should a list of papers or a pop up indicating that there are no such paper	Functional testing	Critical	Pass
T02	Searching authors	Testing if we get list of authors when searching for authors	-.	Search different keywords when search type is Author	There should a list of authors or a pop up indicating that there are no such author	Functional testing	Critical	Pass
T03	Check if layout works	Testing if the layouts runs when we choose different layout types	Have at least 5 nodes and 2 edges in the canvas	Choose a layout option from the layout dropdown	The respective layout should be applied to the graph in the canvas	Functional testing	Minor	Pass

T04	Check if download works	Testing if downloading graphs works	Have at least 1 node in the canvas	Click the download button.	A JSON file should be downloaded.	Functional testing	Major	Pass
T05	Check if uploading a file works	Testing if uploading a JSON file works.	-	Choose a JSON file from the computer and upload it to the canvas using the “Upload file” button	The user should be shown a warning if the file is not JSON format or the corresponding graph should be show in the canvas	Functional testing	Major	Pass
T06	Check if merging new nodes works	Testing if merging newly searched nodes to existing graph works	There should be at least one node in the canvas. The search should not be an empty result.	Search for a new node. Choose some of them from the list. Choose “Merge to Graph” click the “Add” button.	The new nodes should be added to the existing graph in the canvas.	Functional testing	Major	Pass
T07	Check if starting a new graph works	Testing if a new graph starts when choosing to add the	The search should not be an empty result.	Search for a new node. Choose some of them from the list.	The old graph should be removed from the canvas and the new nodes	Functional testing	Minor	Pass

		newly searched nodes to a new graph.		Choose “Start a Graph” click the “Add” button.	and their edges should be added.			
T08	Check if bring references works when adding new nodes	Testing if brining references of newly searched nodes work when adding them to the canvas	There should be nodes that have references in result list	Search for a new node. Choose some of them from the list. Choose “References” and choose distance from drop down. Click the “ADD” button.	The references of the selected nodes should also be added to the canvas as paper nodes.	Functional testing	Major	Pass

T09	Check if bring citations works when adding new nodes	Testing if brining papers that refer of newly searched nodes work when adding them to the canvas	There should be nodes that have citations in result list	Search for a new node. Choose some of them from the list. Choose “Citations” and choose distance from drop down. Click the “ADD” button.	The referring papers of the selected nodes should also be added to the canvas as paper nodes.	Functional testing	Major	Pass
T10	Checking if getting node detail works.	Test if we get the node details when we click on it	There must be at least one paper and at least one author node in the canvas.	Click on the node on the canvas.	The node details of the selected nodes should be shown in the “Node detail” tab of the drawer.	Functional testing	Critical	Pass
T11	Checking if getting	Testing if getting the	There must be at least one author	Click on an author node. Click on the	The papers of the selected author	Functional testing	Critical	Pass

	papers of author works	papers of the author node works	node in the canvas.	“Papers” button.	should be added to the canvas as paper nodes also the edges between these paper nodes and the selected author nodes should be added.			
T12	Checking if removing author node works	Testing if removing the selected author node works	There must be at least one author node in the canvas.	Click on an author node. Click on the “Remove” button.	The selected author node should be removed from the canvas.	Functional testing	Major	Pass
T13	Checking if pinning author node works	Testing if pinning author node works	There must be at least one author node in the canvas.	Click on an author node. Click on the “Pin” button.	The selected author node should be pinned. When a filter is applied it should not affect this pinned node.	Functional testing	Major	Pass

T14	Checking if getting reference s of paper node works	Testing if getting references of selected paper node works	There must be at least one paper node in the canvas.	Click on a paper node. Click on the “References” button.	The references of the selected paper node should be added to the canvas as paper nodes along with the edges between them.	Functional testing	Critical	Pass
T15	Checking if getting citations of paper node works	Testing if getting citations of selected paper node works	There must be at least one paper node in the canvas.	Click on a paper node. Click on the “Citations” button.	The citation of the selected paper node should be added to the canvas as paper nodes along with the edges between them.	Functional testing	Critical	Pass
T16	Checking if getting authors of chosen	Testing if getting authors of selected paper node works	There must be at least one paper node in the canvas.	Click on a paper node. Click on the “Authors” button.	The authors of the selected paper node should be added to the canvas as author	Functional testing	Critical	Pass

	paper works				nodes along with the edges between them.			
T17	Check if removing a paper node works	Testing if removing the selected paper node from the canvas works	There must be at least one paper node in the canvas.	Click on a paper node. Click on the “Remove” button.	The selected paper node should be removed from the canvas.	Functional testing	Major	Pass
T18	Checking if pinning a paper node works	Testing if pinning a paper node works	The database must be running too.There must be at least one paper node in the canvas.	Click on a paper node. Click on the “Pin” button.	The selected paper node should be pinned. When a filter is applied it should not affect this pinned node.	Functional testing	Major	Pass
T19	Check if time filter works	Testing of the time filter works on the Paper nodes existing in the canvas	Have at least 3 nodes in the canvas	Move the two endpoints around to test different time intervals.	The canvas should only display Paper nodes published within the given time interval.	Functional testing	Major	Pass

T20	Checking if filter by area works (and option)	Testing if filter by area works real time for paper nodes when the the filter option is “and”	There must be at least one paper node in the canvas.	Click on the Filter tab in the drawer. choose “and” option from switch. Choose areas of study to consider by clicking on the keywords.	The paper nodes in the canvas should only be paper nodes who have all the selected area of study	Functional testing	Major	Pass
T21	Checking if filter by area works (or option)	Testing if filter by area works real time for paper nodes when the the filter option is “or”	There must be at least one paper node in the canvas.	Click on the Filter tab in the drawer. choose “or” option from switch. Choose areas of study to consider by clicking on the keywords.	The paper nodes in the canvas should only be paper nodes who have at least one of the selected area of study	Functional testing	Major	Pass
T22	Check if filter by citation	Testing if filter by citation count works for paper nodes	There must be at least one paper node in the canvas.	Click on the Filter tab in the drawer. Write min and max number of	The paper nodes in the canvas should only be paper nodes with citation	Functional testing	Major	Pass

	count works			citation counts in the respective text fields. Click on “Filter” button	count within the given range.			
T23	Check if bring common references of selected nodes work	Testing if bringing common references of selected nodes work	There must be at least 2 paper nodes in the canvas.	Click on the “Find Common” button above the canvas. Select some paper nodes. Click on “Bring references” button	The common referenced papers of the selected papers should be added to the canvas as paper nodes along with the edges between the selected node and new nodes.	Functional testing	Critical	Pass
T24	Check if bring common referring papers of selected	Testing if bringing common referring papers of selected nodes work	There must be at least 2 paper nodes in the canvas.	Click on the “Find Common” button above the canvas. Select some paper nodes. Click on “Bring referring papers”	The common referring papers of the selected papers should be added to the canvas as paper nodes along with the edges	Functional testing	Critical	Pass

	nodes work			button	between the selected node and new nodes.			
T25	Check if bring common paper of selected author nodes work	Testing if bringing common paper of selected author nodes work	There must be at least 2 author nodes in the canvas.	Click on the “Find Common” button above the canvas. Select some author nodes. Click on “Bring common papers” button	The common papers of the selected authors should be added to the canvas as paper nodes along with the edges between the selected author node and new paper nodes.	Functional testing	Critical	Pass
T26	Performance of author search	Test the performance of search for author names.	Make sure there are at least 50 author names or aliases that contain the same string which has	Chooses Author as the search type Enter the string to search.	The results should be shown within 2 seconds	Performance	Major	Pass (1+1+2+2+1)ms/5 = 1.4ms

			more than three characters in the database.	See the results.				
T27	Performance of paper search	Test the performance of search for paper titles.	Make sure there are at least 50 paper titles that contain the same string which has more than three characters in the database.	Choose Paper as the search type. Enter the string to search. See the results.	The results should be shown within 2 seconds	Performance	Major	Pass (3+1+1+1+1)ms/5 = 1.8 ms
T28	Performance of adding new nodes from the search	Test the performance when new nodes and edges are added to canvas		Make a search by paper or author Choose ten of them from the result Click Add button	The new nodes and their relations should be added to the canvas in 3 seconds.	Performance	Major	Pass (1 + 1.4 + 1.2 + 0.8 + 1.2)ms/5 = 1.12 ms

				See that they are added to canvas with edges				
T29	Performance of adding references of a paper	Test the performance when references of a paper are added	Have a canvas with at least 20 nodes and at least one them must be a paper with two references which are not on the canvas	Click on a paper node Click on the References button in Node Detail	The three reference papers should be added to the canvas with connecting edges in 2 seconds.	Performance	Major	Pass (1.3 + 1 + 1.2 + 1.2 + 1.1)ms/ 5 = 1.16 ms
T30	Performance of adding citations of a paper	Test the performance when citations of a paper are added	Have a canvas with at least 20 nodes and at least one them must be paper with three citations which are not on the canvas	Click on the paper node Click on the Citations button in Node Detail	The three citation papers should be added to the canvas with connecting edges in 2 seconds.	Performance	Major	Pass (1.2 + 1.3 + 1.3 + 1 + 1.2)ms/5 = 1.2 ms

T31	Performance of adding authors of a paper	Test the performance when authors of a paper are added	Have a canvas with at least 20 nodes and at least one them must be a paper with three authors which are not on the canvas	Click on the paper node Click on the Authors button in Node Detail	The three authors of the paper should be added to the canvas with connecting edges in 2 seconds.	Performance	Major	Pass (1 + 1.3 + 1.3 + 1.3 + 1.5)ms/5 = 1.28ms
T32	Performance of adding papers of an author	Test the performance when papers of an author are added	Have a canvas with at least 20 nodes and at least one them must be an author with two papers which are not on the canvas	Click on the author node Click on the Papers button in Node Detail	The two papers of the author should be added to the canvas with connecting edges in 2 seconds.	Performance	Major	Pass (1.3 + 1.3 + 1.2 + 0.9 + 1.1)ms/5 = 1.16 ms
T33	Performance of finding common references	Test the performance when finding common references	Have a canvas with at least 20 nodes and at least three of them must be paper that have a common	Click Find Common Button Select the three papers	The common reference paper should be added to the canvas with connecting edges in 2 seconds.	Performance	Major	Pass (1.2 + 1.1 + 1.2 + 1.1 + 1.1)ms/5 = 1.14 ms

			reference which is not on the canvas	Click on References Button				
T34	Performance of finding common citations	Test the performance when finding common citations	Have a canvas with at least 20 nodes and at least three of them must be paper that have a common citation which is not on the canvas	Click Find Common Button Select the three papers Click on Citations Button	The common citation paper should be added to the canvas with connecting edges in 2 seconds.	Performance	Major	Pass (1.3 + 1.1 + 1.4 + 1.2+ 1.1)ms/5 = 1.22ms
T35	Performance of finding common papers	Test the performance when finding common papers	Have a canvas with at least 20 nodes and at least three of them must be author that have a common paper which is not on the canvas	Click Find Common Button Select the three authors Click on Papers Button	The common paper of three authors should be added to the canvas with connecting edges	Performance	Major	Pass (1.1 + 1.4 + 1.2 + 1.4 + 1.1)ms/5 = 1.24ms

T36	Performance of Layout	Test the performance of applying the layout	Make sure there are 100 nodes on the canvas	Apply Cose-Bilkent layout Apply Cola layout Apply Dagre layout Apply Klay layout Apply Eular layout	Every layout should be applied within 5 seconds	Performance	Major	Pass (2 + 4 + 1.5 + 1.5 + 2.3)ms/5 = 2.26ms
T37	Performance of Filter	Test the performance of applying filters	Make sure there are 100 nodes on the canvas	Select a start time as a filter Select a start and end time as a filter Select a topic for filter	After every selection for filter, the filter should be applied within 0.5 second	Performance	Minor	Pass (0.3 + 0.5 + 0.4 + 0.4 + 0.7)ms/5 = 0.46ms

				Select second topic for filter Make “and” “or”				
T38	Performance of Node Details	Test the performance of showing details of a node	Make sure there are 100 nodes on the canvas and at least one of time is author and another one is paper	Click on an author node See the details of the node Click on a paper node See the details of the node	After clicking on a node, details should be shown within 0.5 seconds	Performance	Major	Pass (1+1+1+1+1)ms/5 = 1ms
T39	Performance of the interactivity of the Canvas	Test the interactivity of the Canvas	Make sure there are 100 nodes and at least 100 edges on the canvas	Hold and move a node that has at least two edges	The nodes should move with no delays.	Performance	Minor	Pass (1+1+1+1+1)ms/5=(1ms)

T40	Performance of Downloading Canvas	Test the performance of downloading the canvas	Make sure there are 100 nodes and 50 edges on the canvas	Click the download button	The time between clicking the button and starting to download should not be more than 2 seconds	Performance	Minor	Pass (2.9 + 1.5 +1.1+ 2.5+ 0.9)ms/5 =1.78 ms
T41	Performance of Uploading Canvas	Test the performance of uploading the canvas	Make sure there is a downloaded canvas with 100 nodes and 50 edges	Click on the upload canvas button Select the already downloaded canvas Click on the upload button	The canvas should be constructed within 5 second	Performance	Minor	Pass (1.3+1.29+1.19+2.3+2.25)ms/5 =1.66 ms
T42	Correct Edges	Test whether nodes are connected correctly	Make sure there is a canvas that includes both kinds of nodes and edges	Check the papers to see whether the author and reference-of	The data provided by canvas should be correct	Reliability	Critical	Pass

				edges are shown correctly in the canvas				
T43	Correct Node Details	Test the shown detail about a node is correct	Make sure to have a graph that has both kinds of nodes	Click on both kinds of nodes Check the shown details of the node are really the details of clicked node	Details and the clicked node should match	Reliability	Critical	Pass
T44	Correct Queries	Test the queries for a node are correct	Make sure there are both kinds of nodes on the canvas	Click on a paper node Run the query options Click on an author node Run the query options	The results of running queries are correct	Reliability	Critical	Pass

T45	Correct Query options	Test the queries for a node are correct	Make sure there are both kinds of nodes on the canvas	Click on a paper node Run the query options Click on an author node Run the query options	When clicking on different kinds of nodes, query options differ according to that	Reliability	Major	Pass
T46	Download	Test whether download button works fine		When there is no nodes and edges on the canvas try to click on download button When there are both nodes and edges try to click on download button	When there is no nodes and edges, an alert should be shown When there are edges and nodes the canvas should be downloaded without any alert	Usability	Minor	Pass

T47	Upload	Test whether upload button works fine		When the canvas is empty, the upload button is clicked. When the canvas is not empty, the upload button is clicked.	When the canvas empty, upload button should work without an alert When the canvas is not empty, a confirmation message should be shown since the nodes in the canvas will be lost	Usability	Minor	Pass
T48	Search String Length	Test that a search can be done with at least three characters		Enter only one character and try to click the search button. Enter only two characters and try to click the search button.	For the first two trials, the button should not be clickable. For the last trial, the button should be clickable	Usability	Minor	Pass

				Enter only three character and try to click search button				
T49	Select All Button	Testing if all the nodes in the search result list are selected when “Select all” button is clicked	Make a search and have at least five results	Select at least two results Click Select All button	All the items in the search result list should be selected	Usability	Minor	Pass
T50	Selecting None Button	Testing if none of the nodes in the search result list are not selected when “Select none” button is clicked	Make a search and have at least five results	Select at least two results Click Select Nonw button	All the items in the search result list should be unselected	Usability	Minor	Pass

T51	Clickable Add Button	Test the add button after search	Make a search and have a list of results.	Not select any node from the search list and try to click the add button. Select at least one node from the list and click on add button	For the first trail, add button should not be clickable, for the second trial, the add button should work fine and nodes should be added to canvas	Usability	Minor	Pass
T52	Time Filter	Test the time filter	Canvas has more than one paper nodes	Choose a start time. Click the filter button. Choose an end time that is after the start time. Click the filter button. Delete start time.	Apart from the last trial, the filter button should work fine. For the last trial, it should nat be allowed that start time is after the end time.	Usability	Major	Pass

				Click the filter button. Choose a new start time that is after end time. Click the filter button.				
T53	Clickable URLs	Test the URLs in node details are clickable	Have a canvas with some paper and author nodes	Click on the nodes. Click on the URLs in node details.	Another section for the clicked URL should be opened.	Usability	Minor	Pass

6. Maintenance Plan and Details

In our codebase, the endpoint, function, and variable names are descriptive and easy to understand. Additionally, the code is written in a way to separate different tasks into different modules so that functions can be modified without affecting each other too much and new functions can be added easily to maintain an application over time. Therefore, if at any point we get new developers, they would be able to contribute to the project fairly easily as it would not be very challenging for them to understand the codebase. This is one aspect of maintainability.

We will also monitor the feedback given on our website and the issues opened on the Github repository of the project to learn about the bugs once a week. Then, we will fix the bugs.

7. Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

Public Health: PaperAtlas may help medical researchers make more efficient research and thus contribute to the overall public health.

Public Safety: PaperAtlas does not directly affect the safety of its users.

Public Welfare: PaperAtlas does not directly affect the welfare of its users.

Global Factors: PaperAtlas is planned to be a software used by people all around the world. However, people speak many different languages all around the globe. Considering that most people that browse the internet speak English, having the interface in English allows people from different countries to access the software.

Cultural Factors: PaperAtlas may affect the privacy of researchers by making it easier for adversaries to collect data about the researcher’s career. These privacy issues were taken into consideration while building this application.

Social Factors: PaperAtlas may reveal academic groups that gain credit by only citing each other's papers. These groups are also known as academic clans. PaperAtlas may put the social status of such academics at risk.

Environmental: PaperAtlas is a web application, and may not have a direct effect on the environment.
Economic: PaperAtlas does not directly affect the economical situation of its users.

Table 7.1: Factors that can affect analysis and design

	Effect level	Effect
Public health	2	Can help medical research
Public safety	0	-
Public welfare	0	-
Global factors	2	Interface should be in English
Cultural factors	5	Privacy and Data Protection
Social factors	8	Risk of exposing academic dishonesty
Environmental	0	-
Economical	0	-

7.2 Ethics and Professional Responsibilities

While we were developing Paper Atlas we gave importance to ethics and professional responsibilities. Every member of the Paper Atlas team was respectful and helpful to each other throughout the year. Each member of the Paper Atlas team fulfilled the responsibility given to them to implement the application and each member helped each other when there was help needed. Paper Atlas does not use any cookies to collect any information from its users. Also, Paper Atlas does not collect and use any personal data from its users. Therefore it does not share any information with third parties. As a research tool, our main responsibility is to give correct information about papers and authors. We got our papers from Semantic Scholar which is a reliable source with more than 200 million papers. If there is a piece of false information or an author does not want his publicly available paper to be included in Paper Atlas, we are open to making needed changes. In addition, it is our professional responsibility to allow users to understand how the application is used. So we tried to make it user-friendly and we tried to explain Paper Atlas to users in the manual.

7.3. Teamwork Details

After forming our group each team member was assigned a project management task accordingly:

Ahmet Hakan Yılmaz - Managing the project repository and deadlines.

Akın Kutlu - communication with external factors. For example, contacting third party contributors.

Aybala Karakaya - Arranging purposeful meetings and keeping track of time.

Selbi Ereshova - Keeping a track of reports to work on.

Zehra Erdem - Managing the project and checking on work done.

This distribution of responsibility allowed us to overcome many difficulties we had to sustain the team and achieve high quality work. Such division of work eliminated any possible miscommunication and misconduct during the team work. However, dividing the responsibilities does not mean that the members were not

responsible for other parts of the project. The purpose of such division was to guide the team to successfully manage the responsibilities.

Google drive collaboration tool was being used to share the project reports among the team members and allow interaction between us. We were also using Visual Paradigm to simultaneously work on diagrams of our project. Furthermore, GitHub Git tool was used to share the codebase among the team members and keep track of code commits each member makes.

7.3.1. Contributing and functioning effectively on the team

The key ingredient in any team success is the culture of the team. We as a team have been together for more than 1 year and we have built very effective team dynamics which we leverage in our senior design project also. Therefore, so far every member has been making a significant contribution and playing an important role in the project.

Ahmet Hakan Yılmaz:

Hakan made great contributions in managing the necessary repositories and keeping our report web page up to date. Apart from management roles, Hakan developed the responsive and user friendly frontend of our project along with Zehra and Selbi. He worked really hard to build the logic of the frontend. He made great contributions by ensuring the smooth flow of our user interface.

Akın Kutlu:

Akın has done a great job in communicating with the external sources of our project such as third party organization. Akın has reached out to ScholarAPI and persuaded them to give us access to all their data. With Akın's hard work we have more than enough data in our database. He also managed the database by writing scripts to batch process the data in our database. He also contributed to the backend side of the project along with Aybala and Selbi. He also managed to create a bigger database so that we have more connected nodes in our Neo4j database.

Aybala Karakaya:

Aybala did a great job in arranging purposeful meetings and making sure we are making meaningful progress throughout the meetings. She also contributed in developing the backend of our application along with Akın and Selbi. She had made significant contributions in building the endpoints of our system and writing optimized Cypher queries.

Selbi Ereshova

Selbi made great contributions by keeping track of the upcoming reports and ensuring correct formalization and submission of our reports. She was also part of the backend team along with Aybala and Akın. Selbi made great progress in writing endpoints and Cypher queries to retrieve the necessary data from our database and pass it to our frontend. She also contributed to developing better UI for our project along with Hakan and Zehra.

Zehra Erdem:

Zehra did a great job in managing the team progress and making sure everyone is making enough progress. She was also part of the frontend team and had been working with Hakan and Selbi to build a responsive and scalable user interface for our project.

7.3.2. Helping creating a collaborative and inclusive environment

To ensure a collaborative environment, we made use of Zoom meetings, face-to-face meetings, Google Drive, GitHub and WhatsApp. The Zoom meetings and face-to-face meetings helped us to collaborate synchronously, Google Drive and GitHub asynchronously, and WhatsApp was helpful in both.

For all assignments, we performed work division under the leadership of Zehra Erdem. Therefore, every assignment was completed with collaboration. Additionally, when dividing the work, we sometimes assigned one task to more than one team member. This helped us complete those tasks by collaboration. When someone on the team has trouble working on the task, they asked for help on our WhatsApp group. Then, other team members helped them either on WhatsApp, by arranging a Zoom call or a face-to-face meeting.

To create an inclusive environment, we always supported each other by answering each others' questions, motivating other team members, and by being respectful in communication.

The summary of every team members' distinct role in helping creating a collaborative and inclusive environment is as follows:

Ahmet Hakan Yılmaz:

He created the project repository which is a key component of our coding collaboration. Also, while managing the deadlines he was mindful of the team members' other commitments which helped us create an inclusive environment.

Akın Kutlu:

He led the data collection and database creation. While doing so, he was a respectful lead who made sure every team member felt included. He also worked mostly independently while communicating with third parties, such as the managers of Semantic Scholar API. During this task, he periodically informed the group of the progression of the work and asked for help and support when he needed it which helped us to get collaboration.

Aybala Karakaya:

She arranged and runed the meetings we have regularly and that was a key factor in having collaboration. While running the meetings, she made sure that everyone gets an opportunity to speak which helped us get an inclusive environment.

Selbi Ereshova:

She kept a track of reports to work on. To do this, she created Google Docs files which we store on Google Drive. That gived us the opportunity to easily collaborate asynchronously. Additionally, while leading the work for First Demo, she was an inclusive lead.

Zehra Erdem:

She managed the project and checked on the work done. Since she also led the process of work division, we were able to collaborate easily. When dividing the work, she made sure that everyone gets an equal chance to work on impactful tasks. Therefore, she helped us obtain an inclusive environment.

7.3.3. Taking lead role and sharing leadership on the team

Team members shared leadership on the team by taking on the lead role in different tasks. This was done in order to let every team member learn how to take the responsibility of a whole team. The tasks that every team member was responsible for is as the following:

Ahmet Hakan Yılmaz: Detailed Design Report

Akın Kutlu: Collecting Data

Aybala Karakaya: Final Report

Selbi Ereshova: First Demo, Final Demo

Zehra Erdem: Project Specification Report, Analysis and Requirement Report

7.3.4. Meeting objectives

As we progressed through our project some of our objectives changed since the data we acquired was different then what we initially expected. Paper nodes in our database do not include journal attributes. That is why we removed grouping by journal functionality from our objectives. The same situation applies to searching by journal and university since none of these attributes were present in the dataset we found.

Also, we realized that having keywords as nodes alongside Paper and Author makes the UI more complicated to understand. That is why we decided to keep the keywords for filtering purposes only.

However we did implement many other new features such as real time filtering papers by keywords, citation count, pinning nodes, bringing authors of papers, bringing papers of authors, bringing cited papers of a paper, bringing references of a paper, bringing referred and cited authors of an author, uploading and downloading files.

In terms of non-functional requirements, we have reached all of our objectives. In terms of project goals, we exceeded our expectations.

7.4 New Knowledge Acquired and Applied

While developing Paper Atlas we learned and applied different technologies or frameworks. With some of the frameworks we had an experience but we got more experience such as Javascript, Typescript, and NodeJs. We also used Express.js within our Node.js application. For the frontend, we used React and learned about React hooks and component libraries such as Material UI. Most of the team did not know about graph databases or libraries at first. Therefore we learned about graph-based databases and libraries that can help us in graph visualization. We used Neo4j as a database management system which is a Graph Data Platform. Neo4j is a noSQL database system and we also used Cypher which is the language for the database. Another framework related to graphs we use is Cytoscape.js which allows us to work on graphs. We also used APIs for fetching some of our data from Semantşc Scholar.

8. Conclusion and Future Work

In conclusion, we have developed Paper Atlas which is a web application that visualizes various graphs including academic papers, authors and relations between academic papers. We believe that Paper Atlas could be a very helpful tool for people who want to do research about a topic by guiding its users. With the power of data visualization, Paper Atlas allows users to gain valuable insights into the connections between papers and authors. Moreover, interactivity of the Paper Atlas allows users to navigate the graph, filter data, and explore specific connections of interest.

Although Paper Atlas has 400000 papers, it is still a small number compared to all papers included in Semantic Scholar. One of the future works is to enhance the current specific dataset to a larger scope. Another future work can be adding a register login system. Then people's graph history can be stored and people who want to see their history can see. Last future work can be adding more personalization to Paper Atlas. Users may choose the color theme of Paper Atlas or the color theme of their graphs, or they may arrange a size algorithm for nodes.

9. User manual

9.1 Search

The user can open search by clicking the arrow at the top of right corner and clicking on Search. There are two types of search which are Paper and Author. The user can choose between them as in Figure 9.1.1. If Paper is chosen, the user can enter a keyword to search among paper titles or if Author is chosen, the user can enter the keyword to search among author names. The result will be shown after clicking the Search button in pages. The user can navigate between pages by clicking the numbers or the arrows that are at the end of the list (Figure 9.1.2). The user can manually select some results and can use Select All or Select None buttons. If none of the results are chosen, the Add button will be unclickable (Figure 9.1.2). Before adding selected results, both paper and author search have options which are "Merge to the graph" and "Start a graph" (Figure 9.1.2 and 9.1.3). If users want to keep nodes on the graph and add new ones over them, they should click

Merge. If users want to start a new graph with selected results they can choose the Start option. The default option is Merge. After choosing the option and selecting some results the user can click the Add button to add selected results to the graph as nodes.

For paper type of search, there is an additional part as can be seen in Figure 9.1.3 If the user wants, they can bring citations of selected papers, references of selected papers or both of them as well by choosing References and Citations options. By entering a number as distance, the user can decide how far citations or references will be brought with selected ones.

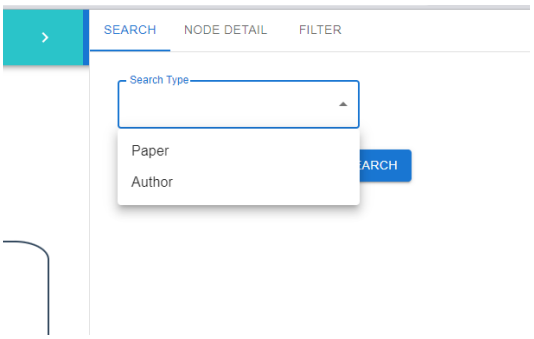

Figure 9.1.1: Search Types

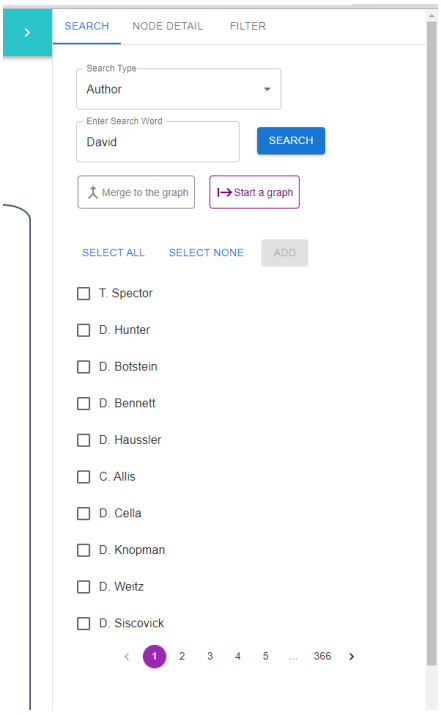

Figure 9.1.2: Author Type of Search

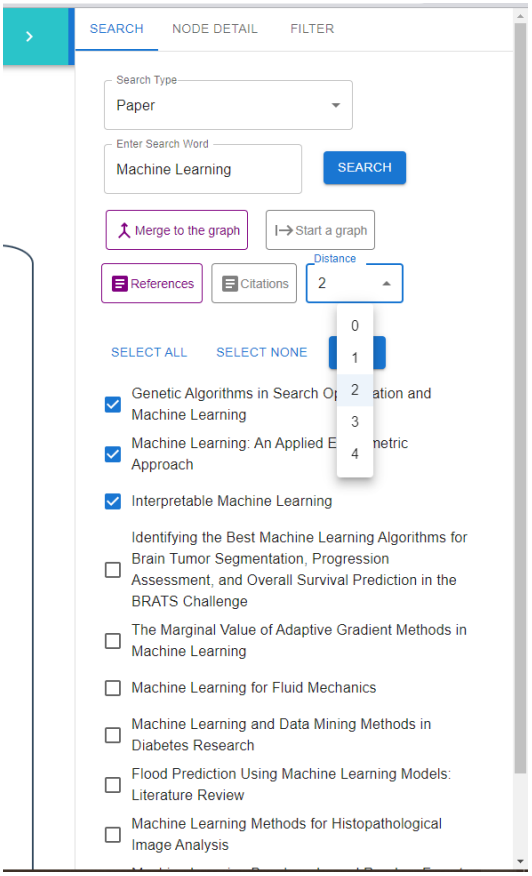

Figure 9.1.3: Paper Type of Search

9.2 Node Details

When a user clicks a node the node details tab on the right side of the page is opened. In this node details tab; if the node is an author then the name of the author, aliases of the author, paper count, and citation count are displayed. For author nodes in the node details tab, besides the given information there are 3 buttons; the papers button brings papers written by the selected author to the graph, the remove button removes the node, and the pin button pins the node and filters does not affect pinned nodes.

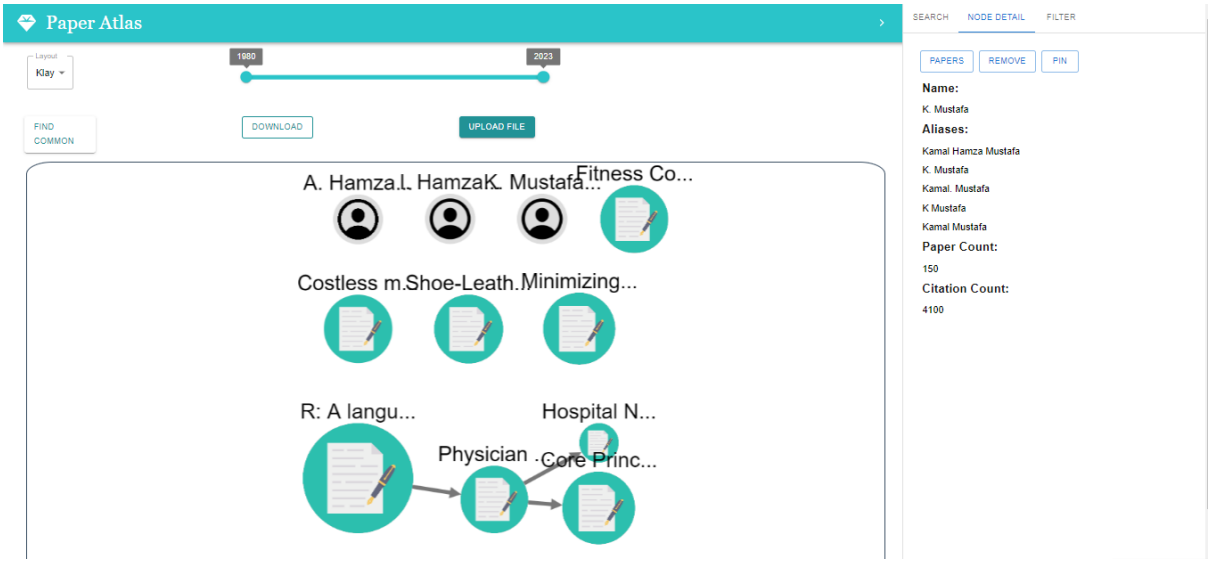

Figure 9.2.1: Node Details of an Author at the right side of page.

Figure 9.2.2: Node Details of an Author

If the node is a paper then the title, publication year, doi of the paper, fields of the paper, a link to the paper, and abstract of the paper are displayed. For paper nodes in the node details tab, besides given information there are 5 buttons, the references button brings papers that are referenced by the source node, the citation button brings papers that refer to the source node and the authors button brings authors of the source paper. The remove button removes the selected node and the pin button pins the node and filters can not affect the pinned node.

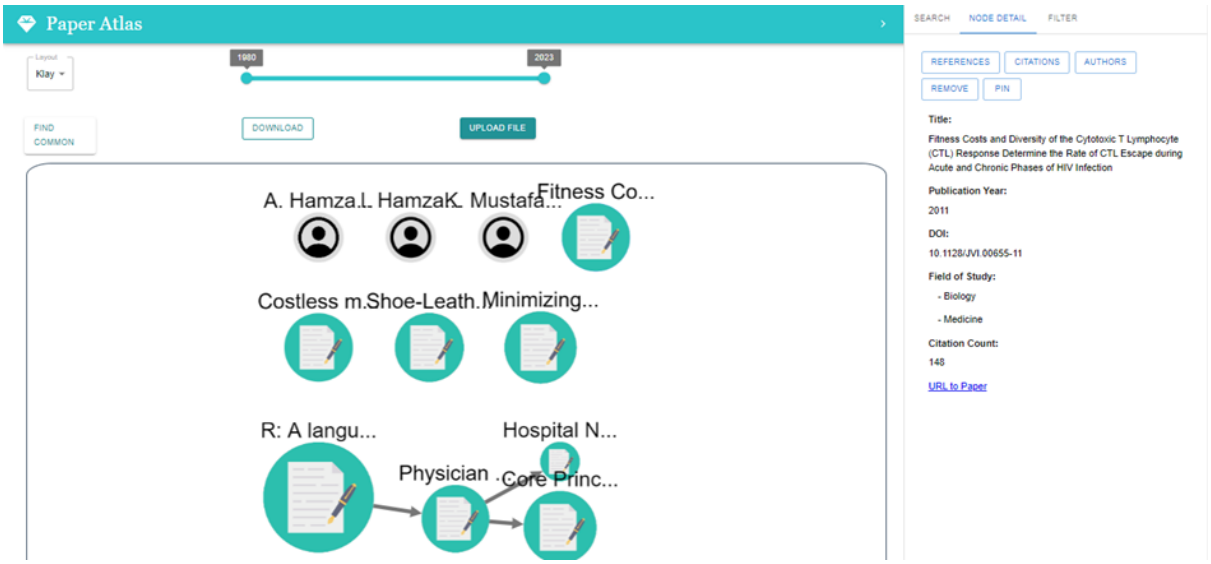

Figure 9.2.3: Node Details of a Paper at the right side of the page.

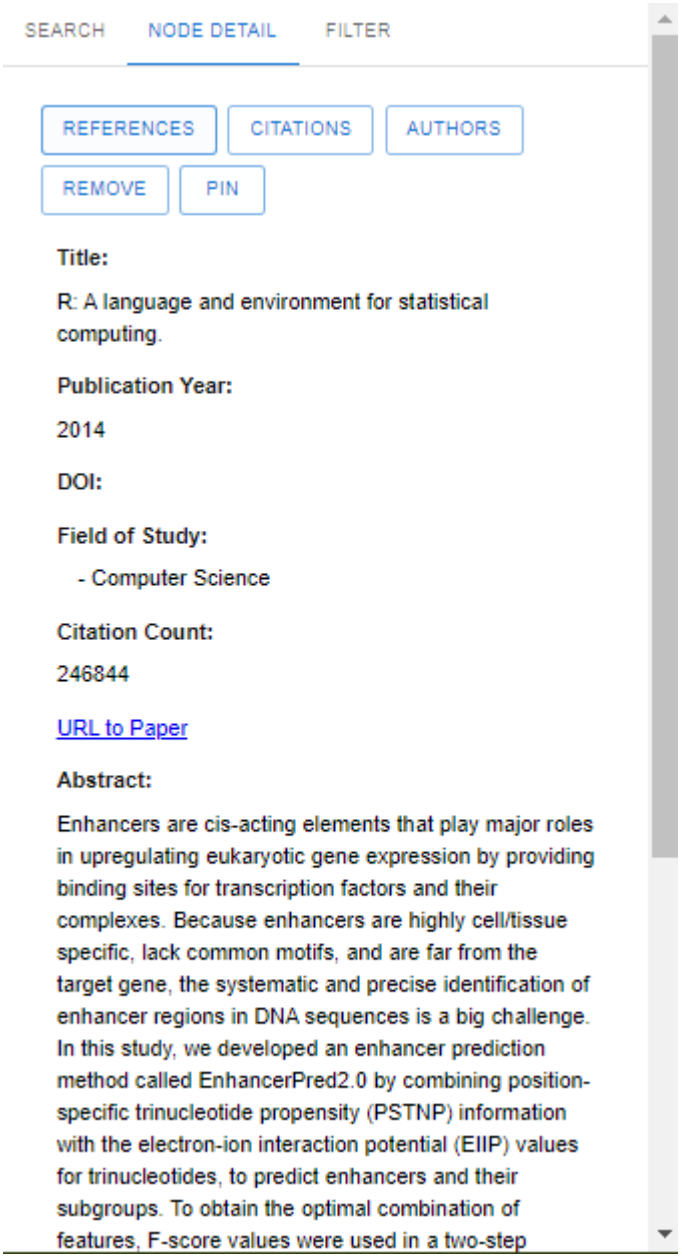

Figure 9.2.4: Node Details of a Paper

9.3 Filter

There are three types of filters: publish year, area, and citation count. The user can filter the nodes on the graph except pinned nodes. The pinned nodes do not affected by any of the filters. The user can change the start and end year with a time filter and the papers which publish the year between the start and end year will be shown. In filter by area, there are hard-coded areas as buttons and the user can select one or more of them at the same time. Also, the user can change the toggle (OR and AND). According to this toggle button, areas will be filtered. The last filter is citation count. The user can write a min and max number of citation count and press apply the filter to eliminate nodes whose citation count is not in between min and max numbers. The filters can be applied together or individually.

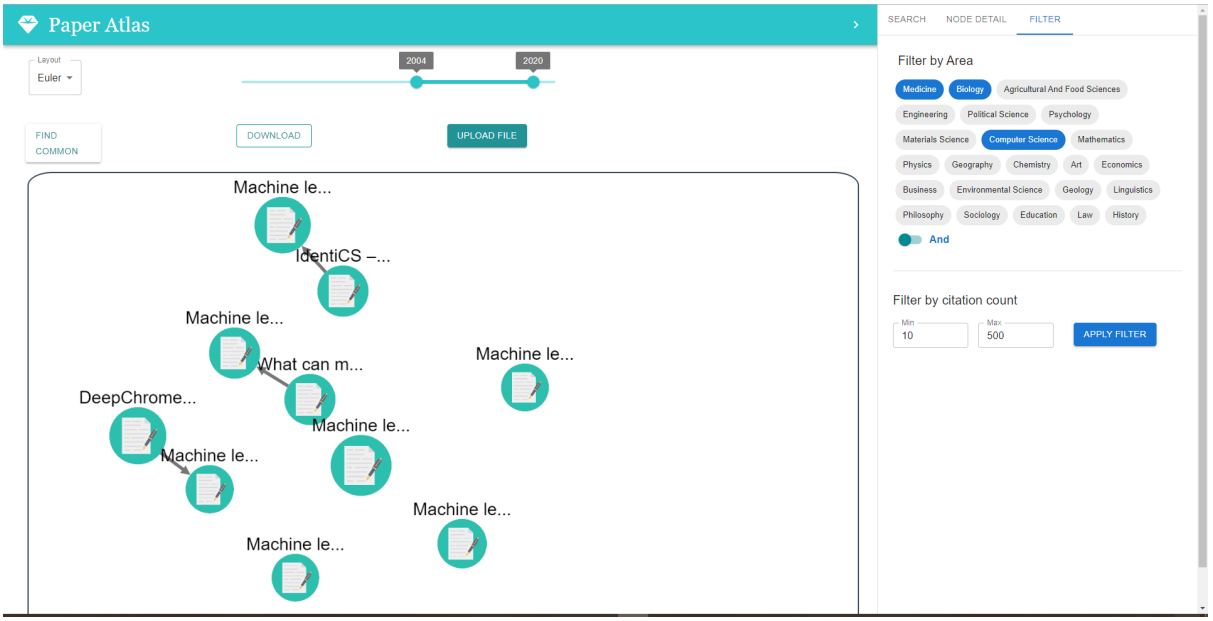

Figure 9.3: Filter By Area Example

9.4 Canvas - Layout

On the graph page, users change the layout of the graph as they like. There are several options for layout: Cola, Cose Bilkent, Dagre, Euler, and Klay. Each one provides a different look and can be useful for different graphs. In order to change the layout, the user can use the layout drop-down menu. Changing layout does not affect filters or pinned nodes, it only changes the location of them. In the canvas, users can interact with author and paper nodes. These nodes can be moved to another place and the related nodes will be still connected.

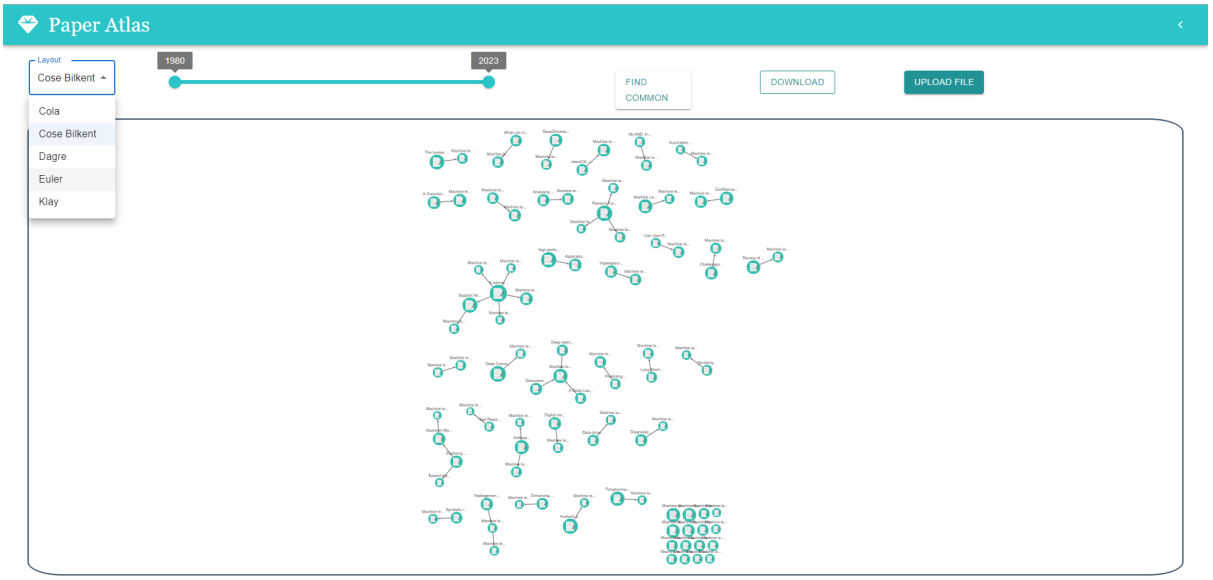

Figure 9.4.1: Layout selection example

Figure 9.4.2: After layout changed example

9.5 Downloading and Uploading

Users can choose to download their graph as a JSON file so they can reuse it later. To do this they can click on the Download button above the canvas.

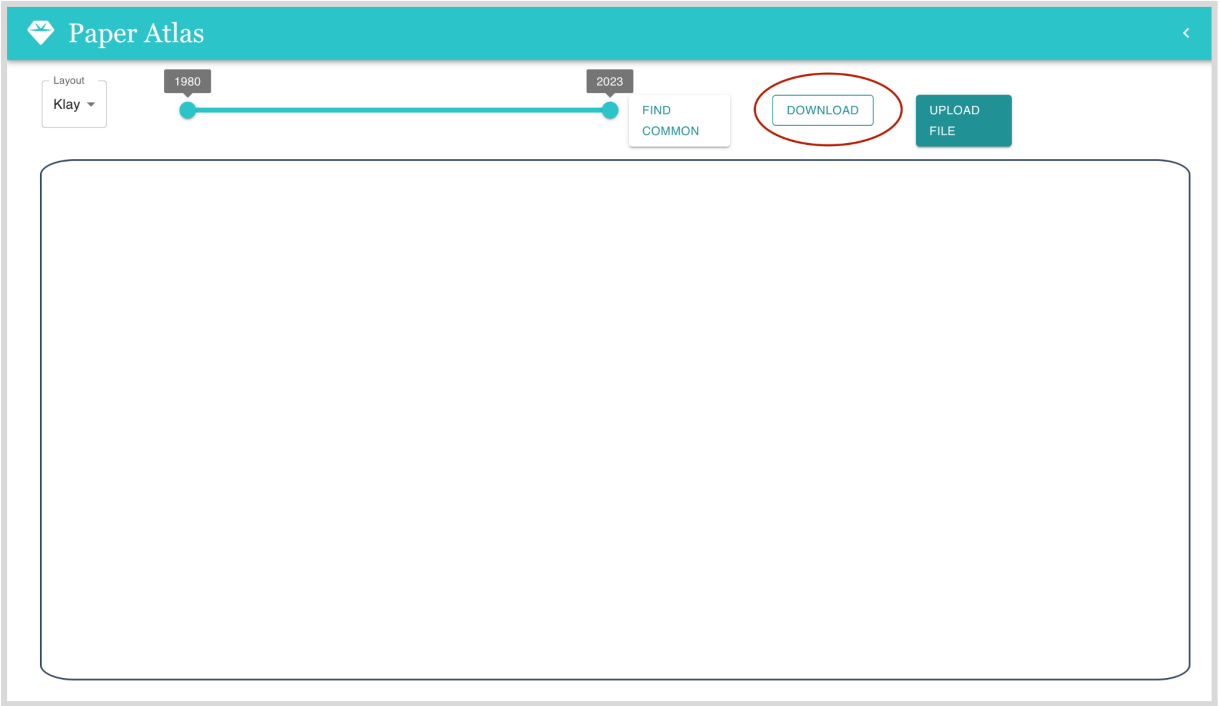

Figure 9.5.1: Downloading the graph in the canvas

When the user clicks “Download”, if the graph is empty they will get an alert warning saying that they cannot download an empty graph:

Users can upload a previously downloaded JSON file to the application by clicking the “Upload file” button above the canvas.

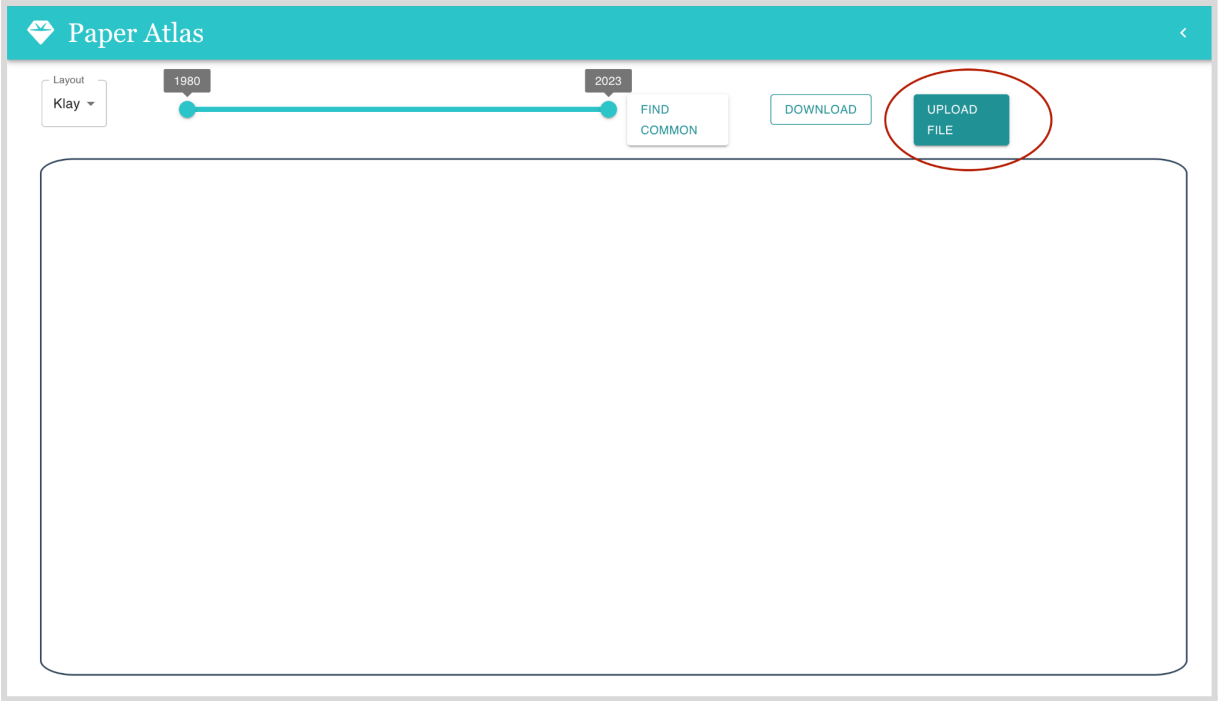

Figure 9.5.2: Uploading a graph to the canvas

The user can choose a file from their computer:

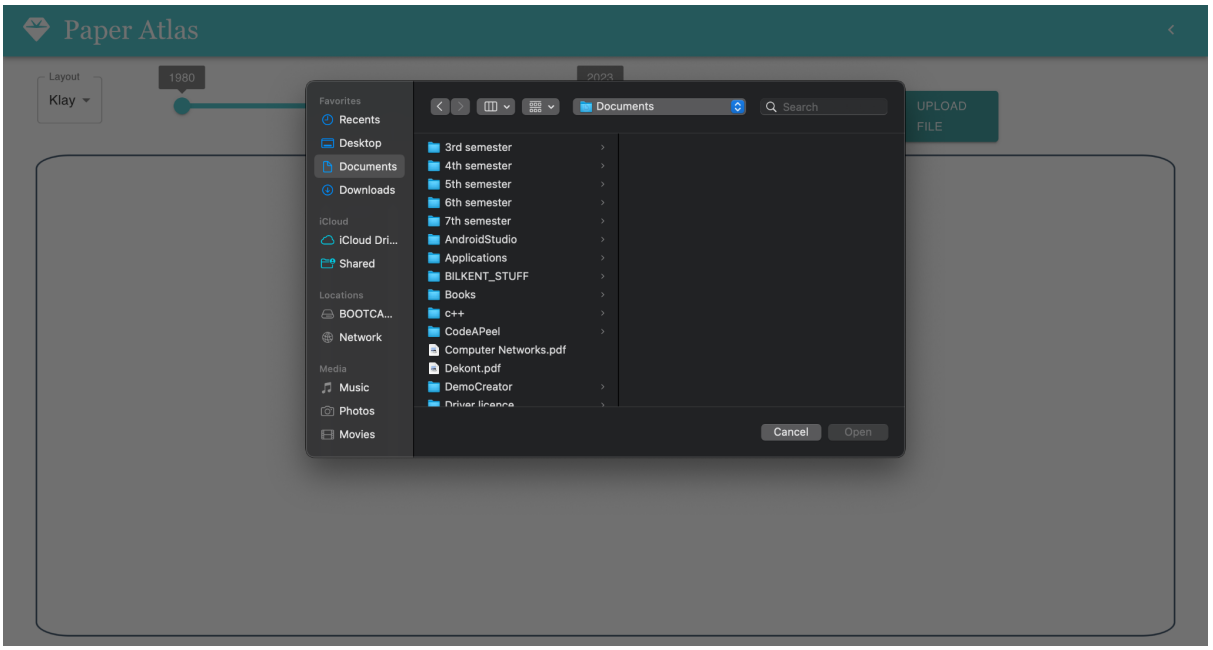

Figure 9.5.3: Choose a file to upload as a graph to the canvas

Before the user can upload the file, our app checks if the file is in JSON format. If not an alert is shown to the user saying “Wrong Format! Please upload JSON files only” and the file is not uploaded to our application.

Also, if there are any nodes present in the canvas, then a pop up is shown asking the user if they are sure they want to replace the current graph with the new uploaded one.

9.6 Finding Common Papers

Users can find common references of some papers they choose, the common papers that are referring some papers they choose, or the common papers of the chosen authors.

Figure 9.6.1: Before Finding Commons

To do that, they must first click on the “Find Common” button”. Then, “Common References”, “Common Citations” and “Common Papers of Authors” buttons will be displayed. “Common References” brings the common papers that the input papers are referencing. “Common Citations” brings the common papers that are referencing the input papers. “Common Papers of Authors” brings the common papers of the input authors.
nodes they want to find a common property of. Then, they must click on the.

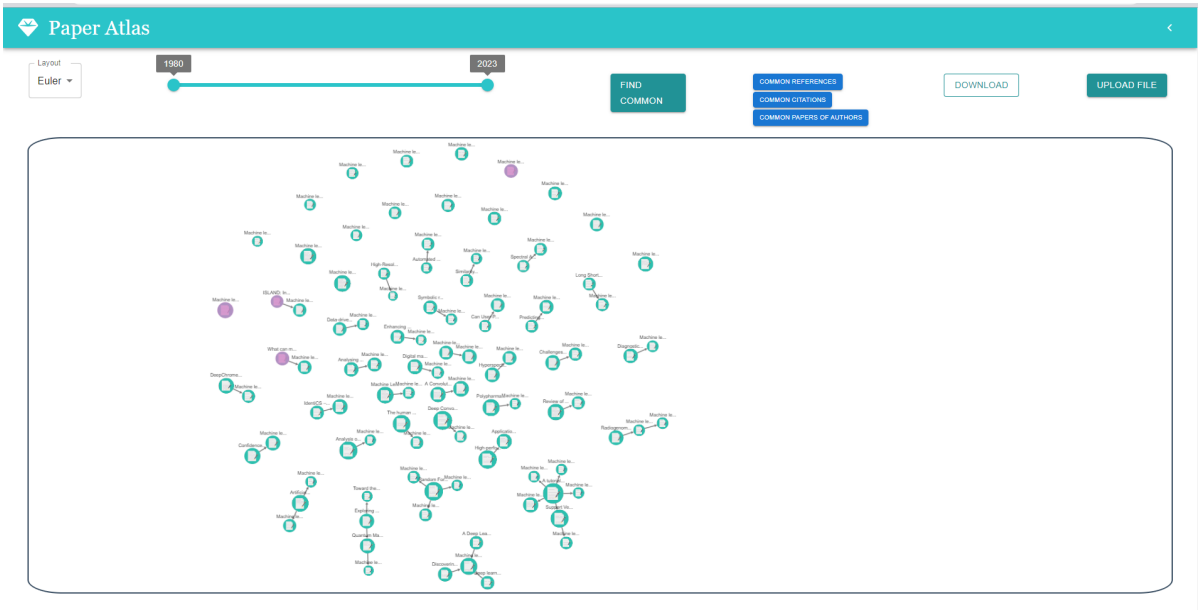

Figure 9.6.2: After Clicking the Find Common

After clicking the “Find Common” button, users must click on the nodes they want to find a common property of. The chosen nodes are displayed in dark blue. When the user clicks on a node to find common properties, it becomes dark blue. Then, they should choose the common property they want to see by clicking on one of the “Common References”, “Common Citations” and “Common Papers of Authors” buttons.

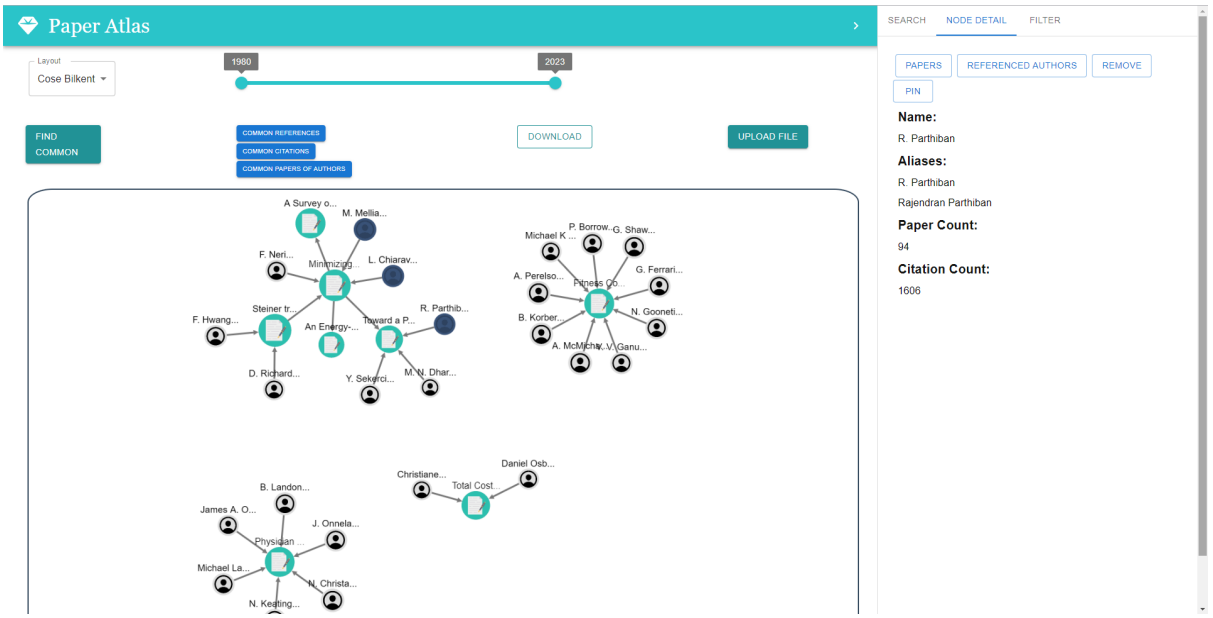

Figure 9.6.3: Selecting Nodes to Find Commons

10. Glossary

Semantic Scholar: It is a platform that helps researchers quickly find papers in their field, it includes features such as personalized recommendations, and filtering options to help users explore and understand the research landscape. Semantic Scholar includes more than 200 million papers [1]. We used semantic scholar to fill our database and scholar api to show some details of papers like abstract.

Neo4j: It is a graph database management system that stores and manages the data in the form of a graph [2]. We used Neo4j as the database system management as we store our data in the graph form.

Cypher: Cypher is a graph query language for Neo4j [3]. We use Cypher on the backend to perform queries on our database.

React.Js: React is a framework that uses node to build user interfaces. Javascript and Typescript can be used in React. Components of React return a renderable object [4]. It is used for frontend implementation.

Cytoscape.js: Cytoscape.js is an open-source JavaScript library for creating and visualizing graphs and networks. Cytoscape.js enable users to create and manipulate graphs in a web browser environment [5]. It is used in the frontend to create Canvas.

Canvas: Canvas is used for the rectangular area in the UI that shows all nodes and edges.

11. References

[1] “AI-Powered Research Tool,” Semantic Scholar, https://www.semanticscholar.org/ (accessed May 10, 2023).

[2] “Neo4j graph database & analytics – the leader in graph databases,” Graph Database & Analytics, https://neo4j.com/ (accessed May 10, 2023).

[3] “Cypher query language - developer guides,” Neo4j Graph Data Platform, https://neo4j.com/developer/cypher/ (accessed May 10, 2023).

[4] React, https://react.dev/ (accessed May 10, 2023).

[5] M. Franz, Cytoscape.js, https://js.cytoscape.org/ (accessed May 10, 2023).