Bilkent University
Department of Computer Engineering

Senior Design Project

T2334

PaperAtlas

Detailed Design 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

13.03.2023

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

1.1 Purpose of the System

With Paper Atlas, users can visualize the relations of research papers or authors as a graph-based and interactive structure in order to find the most relevant paper to their topic. The goal of this project is to enhance 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 related to the input paper of the user.

1.2 Design Goals

1.2.1 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.

1.2.2 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.

1.2.3 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.

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

1.3 Definitions, Acronyms and Abbreviations

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. We used semantic scholar to fill our database. We also aim to use semantic 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. We use Neo4j as the database system management as we store our data in the graph form.

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.

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

1.4 Overview

Paper Atlas will be a web application available on all modern web browsers. Paper Atlas will show 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 will enhance 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 users will be able to read the details such as

name,keywords, and abstract by clicking on the node of a paper. A link to the original paper will be available as well. The graph will be 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, keywords, author, and publication journal. Users can also search authors or keywords instead of paper titles. In this case, relative nodes will be shown for authors or keywords. The application will also provide extended features such as grouping of the nodes and ranking papers by their importance.

2. Alternative Software's Architectures

2.1 Google Scholar

Google Scholar [1] is one of the most known websites to search for academic papers and academic people. It is only available on the internet and there is no mobile version of it. Users can search papers in different languages. In the advanced search part, papers can be searched with several detailed options such as "with all of the words", "with the exact phrase" and "with at least one of the words". Also, articles can be filtered as "authored by", "published in" and "dated between". The results are listed and accessible to the user. Users can also look at their profile, library, alerts, metrics, and settings.

2.2 Litmaps

Litmaps [2] is another website to search for academic papers and visualize results. It also uses graphs to visualize the graphs. Within Litmaps, only papers are used as nodes. New papers can be added to the graph or can be deleted from the graph but the graph is not interactive. The positions of the nodes are fixed. Though reference relations are shown with edges, they do not have directions so it is hard to understand which one refers to the other one. Litmaps also uses filters which are depth, date range and keyword.

3. Proposed Software Architecture

3.1 Overview

In this section, the architectural structure 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, database layer and Third Party API. The whole system is composed of four main parts: Frontend layer, Backend layer, Database layer 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 request comes 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 service has access to database layer and third party api service can get data from third party api.

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 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.

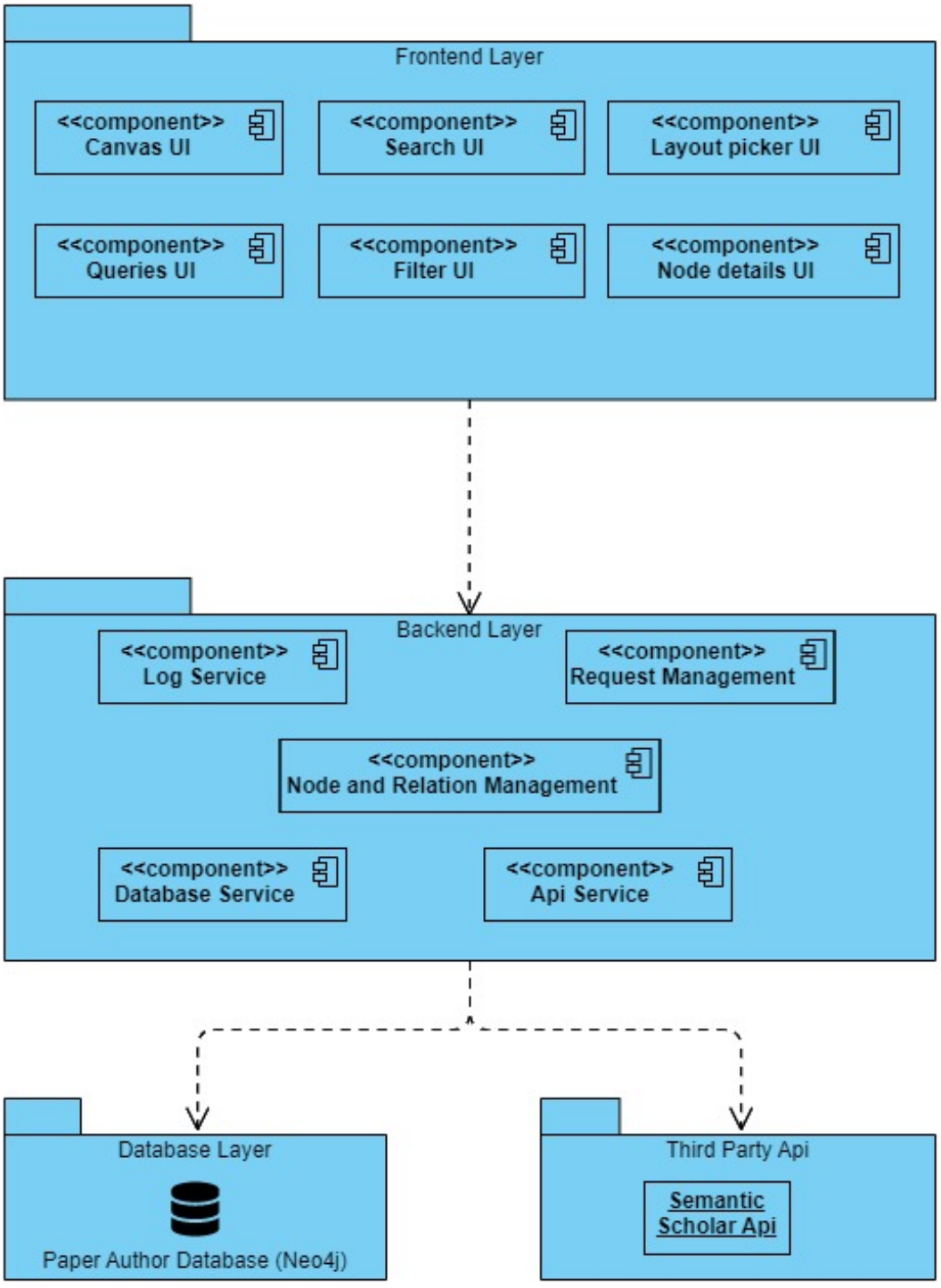

Figure 3.2: Subsystem decomposition diagram

3.3 Hardware/Software Mapping

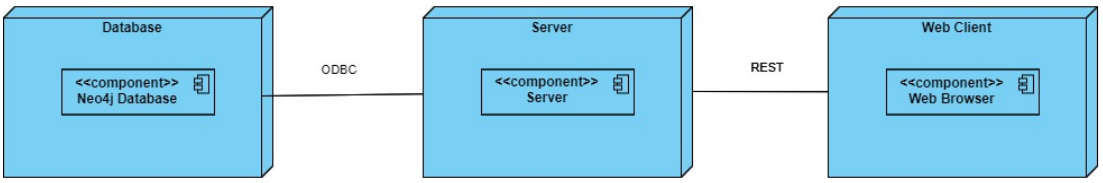

Figure 3.3: Hardware/Software Mapping

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 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 our database

Author nodes: 116587	Paper nodes: 61078	Total nodes: 177665
a-reference-of relations: 14524	an-author-of relations: 151623	Total relations: 166147

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

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.

4. Subsystem Services

Request Management

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

Node and Relation Management

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

Database Service

This module will communicate with Database Server. Within this module queries that will return desired data will be written and run on the Neo4j Database Server.

API Service

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

Log Service

This module will be responsible for tracking the events in the backend layer. It will create and write logs of each event so that when an error occurs, these logs can be used to trace back and find what went wrong.

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.

Queries

Queries component is the component that enables users to enhance the graph from nodes of the graph. For example, with the queries component users can bring references of a node in a 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, by journal, by author, and by topics.

Node details

When a node in the canvas is clicked, 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.

5. Class Diagram

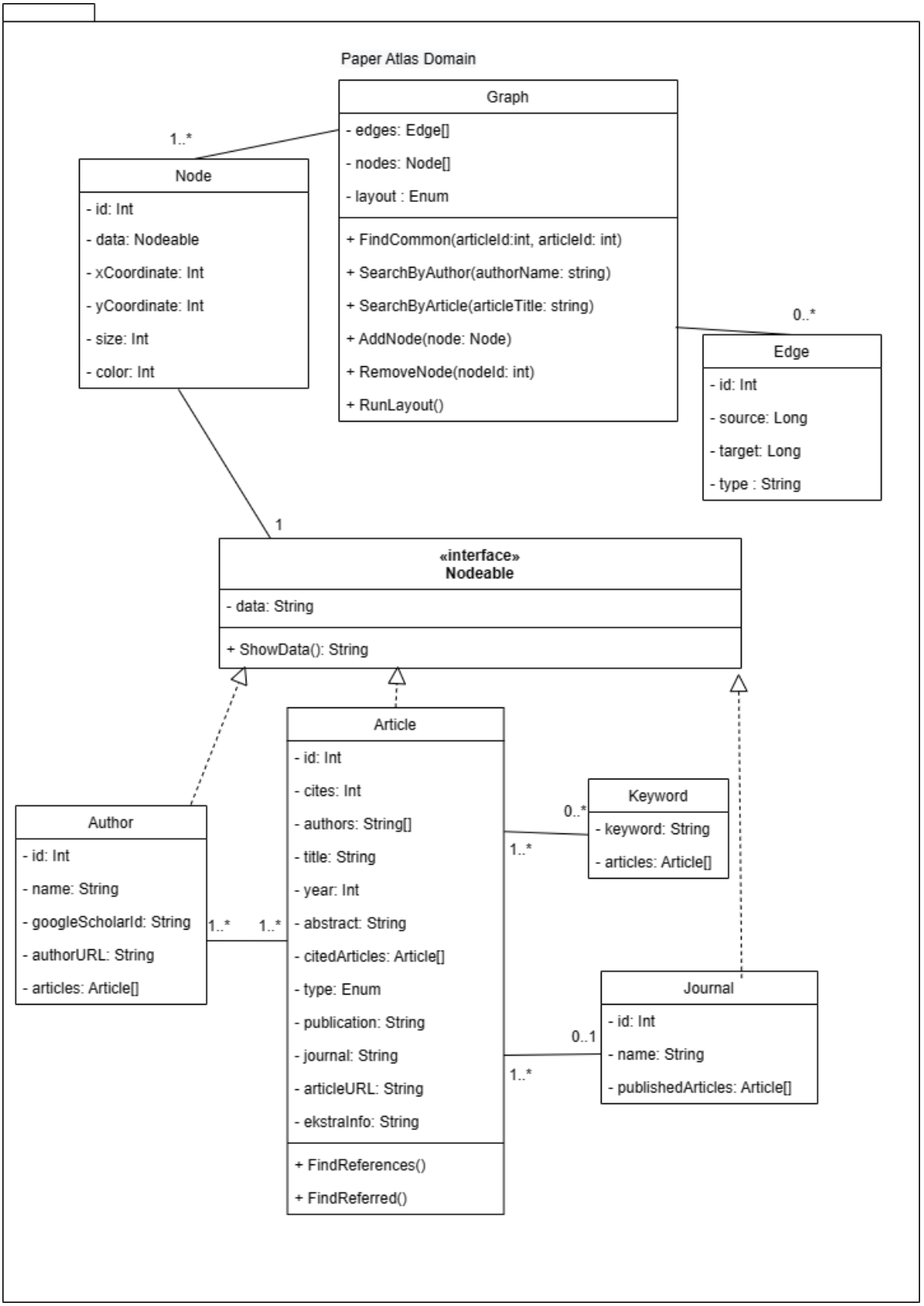

Figure 5: Class Diagram

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 type of edges between different kinds of nodes.

Nodeable

Nodeable class is an interface which is inherited by Author, Article and Journal 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 id, name ,googleScholarId, authorURL and articles properties.

Article

The purpose of this class is to create a respective class Article 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 for getting authors of the article, finding references of the article, and finding referred papers.

Journal

The purpose of this class is to be able to store journal information in the database so that we can filter articles easily in a given journal. This class has a “name” property as a string and “publishedArticles” property as an array of Article objects. “name” is the name of the journal and “publishedArticles” is the articles published in that journal. This class also implements a method to return articles published in that journal.

Keyword

The purpose of this class is to be able to store available keywords in the database so that we can filter articles easily based on related keywords. This class has a “keyword” property as a string and “articles” property as an array of Article objects. “keyword” is the keyword related to that object and “articles” is the articles which are related to that keyword. This class also implements a method to return articles with that keyword.

6. Test Cases

Table 6: Test Cases

Test case name	**Summary**	**Post Procedure**	**Procedure of testing**	**Expected result**	**Category**	**Severity result**
Database connectivity	Testing if the backend is successfully connected to the database		Make a dummy query to the database when deploying the application.	It is expected that the query to the database should not cause any error	Integration testing	Major
Input check for getPaperNames function	Testing if valid string is given as a name input for getPaperNames endpoint	The application is connected to our database. There are at least 50 papers in our database	The getPaperNames endpoint will be called with the “name” parameter.	The input name should be a valid string	Functional testing	Minor
Output check for getPaperNames function	Testing if the output of the getPaperNames	The application is connected to our database. There	The getPaperNames endpoint will call	The output of the query should not be an error.	Functional testing	Major

	query	are at least 50 papers in our database.	a controller to send a query to the database.			
Input check for getAuthorNames function	Testing if the input name of the getAuthorNames endpoint is a valid string	The application is connected to our database. There are at least 50 authors in our database.	The getAuthorNames will be called with the “name” parameter.	The input: name of the getAuthor endpoint should be a valid string	Functional testing	Minor
Output check for getAuthorNames function	Testing if the output of the searchByAuthor query return an error	The application is connected to our database. There are at least 50 authors in our database.	The getAuthorNames endpoint will call a controller to send a query to the database.	The output of the searchByAuthor query should not return an error	Functional testing	Major
Input check for getNeighbor function	Testing if valid string is given as a name input for getNeighbor endpoint	The application is connected to our database. There are at least 50 authors and at least 50 papers in	The getNeighbor endpoint will be called with “limit” and “name” parameters.	The input name should be a valid string	Functional testing	Minor

		our database.				
Input check for getNeighbor function	Testing if valid number is given as a length limit input for getNeighbor endpoint	The application is connected to our database. There are at least 50 authors and at least 50 papers in our database.	The getNeighbor endpoint will be called with “limit” and “name” parameters.	The input length limit input should be a valid number	Functional testing	Minor
Output check for getNeighbor function	Testing if the output of the getNeighborOfPaper query is an error	The application is connected to our database. There are at least 50 authors and at least 50 papers in our database.	The getNeighbor endpoint will call a controller to send a query to the database.	The output of getNeighborOfPaper should not be an error	Functional testing	Major
Input check for GetPapers endpoint	Testing if valid list of integers is given as an input for GetPapers endpoint	The application is connected to our database. There are at least 50 papers in our database.	The GetPapers endpoint will be called with a list of ids as a parameter.	The input list if ids should be a valid array.	Functional testing	Minor

Input check for GetPapers endpoint	Testing if the inputs for GetPapers endpoint are a valid id in our database.	The application is connected to our database. There are at least 50 papers in our database.	The GetPapers endpoint will be called with an id as a parameter.	The input ids should be valid paper ids in our database.	Functional testing	Minor
Output check for GetPapers endpoint	Testing if the output of the GetPapers query is an error	The application is connected to our database. There are at least 50 papers in our database.	The GetPapers endpoint will call a controller to send a query to the database.	The output of GetPapers should not be an error	Functional testing	Major
Input check for GetAuthors endpoint	Testing if valid list of integers is given as an input for GetAuthors endpoint	The application is connected to our database. There are at least 50 authors in our database.	The GetAuthors endpoint will be called with a list of ids as a parameter.	The input list if ids should be a valid array.	Functional testing	Minor
Input check for GetAuthors endpoint	Testing if the inputs for GetAuthors	The application is connected to our database. There	The GetAuthors endpoint will be called with an id	The input ids should be valid author ids in our	Functional testing	Minor

	endpoint are a valid id in our database.	are at least 50 authors in our database.	as a parameter.	database.		
Output check for GetAuthors endpoint	Testing if the output of the GetAuthors query is an error	The application is connected to our database. There are at least 50 authors in our database	The GetAuthors endpoint will call a controller to send a query to the database.	The output of GetAuthors should not be an error	Functional testing	Major
Input check for getReferences endpoint	Testing if valid integer is given as an input for getReferences endpoint	The application is connected to our database. There are at least 50 papers and 25 reference-of relations in our database	The getReferences endpoint will be called with an id as a parameter.	The input ids should be a valid integer.	Functional testing	Minor
Input check for getReferences endpoint	Testing if the input for getReferences endpoint is a valid	The application is connected to our database. There are at least 50	The getReferences endpoint will be called with an id	The input id should be a valid paper id in our database.	Functional testing	Minor

	id in our database.	papers and 25 reference-of relations in our database	as a parameter.			
Output check for getReferences endpoint	Testing if the output of the getReferences query is an error	The application is connected to our database. There are at least 50 papers and 25 reference-of relations in our database	The getReferences endpoint will call a controller to send a query to the database.	The output of getReferences should not be an error	Functional testing	Major
Input check for getReferred endpoint	Testing if valid integer is given as an input for getReferred endpoint	The application is connected to our database. There are at least 50 papers and 25 reference-of relations in our database	The getReferred endpoint will be called with an id as a parameter.	The input ids should be a valid integer.	Functional testing	Minor

Input check for getReferred endpoint	Testing if the input for getReferred endpoint is a valid id in our database.	The application is connected to our database. There are at least 50 papers and 25 reference-of relations in our database	The getReferred endpoint will be called with an id as a parameter.	The input id should be a valid paper id in our database.	Functional testing	Minor
Output check for getReferred endpoint	Testing if the output of the getReferred query is an error	The application is connected to our database. There are at least 50 papers and 25 reference-of relations in our database	The getReferred endpoint will call a controller to send a query to the database.	The output of getReferred should not be an error	Functional testing	Major
Input check for getAuthors endpoint	Testing if valid integer is given as an input for getAuthors endpoint	The application is connected to our database. There are at least 50 papers, 50	The getAuthors endpoint will be called with an id as a parameter.	The input ids should be a valid integer.	Functional testing	Minor

		authors and 25 author-of relations in our database				
Input check for getAuthors endpoint	Testing if the input for getAuthors endpoint is a valid id in our database.	The application is connected to our database. There are at least 50 papers, 50 authors and 25 author-of relations in our database	The getAuthors endpoint will be called with an id as a parameter.	The input id should be a valid paper id in our database.	Functional testing	Minor
Output check for getAuthors endpoint	Testing if the output of the getAuthors query is an error	The application is connected to our database. There are at least 50 papers, 50 authors and 25 author-of relations in our	The getAuthors endpoint will call a controller to send a query to the database.	The output of getAuthors should not be an error	Functional testing	Major

		database				
Input check for getPapers endpoint	Testing if valid integer is given as an input for getPapers endpoint	The application is connected to our database. There are at least 50 papers, 50 authors and 25 author-of relations in our database	The getPapers endpoint will be called with an id as a parameter.	The input ids should be a valid integer.	Functional testing	Minor
Input check for getPapers endpoint	Testing if the input for getPapers endpoint is a valid id in our database.	The application is connected to our database. There are at least 50 papers, 50 authors and 25 author-of relations in our database	The getPapers endpoint will be called with an id as a parameter.	The input id should be a valid author id in our database.	Functional testing	Minor

Output check for getPapers endpoint	Testing if the output of the getPapers query is an error	The application is connected to our database. There are at least 50 papers, 50 authors and 25 author-of relations in our database	The getPapers endpoint will call a controller to send a query to the database.	The output of getPapers should not be an error	Functional testing	Major
Input check for getAuthorDetail endpoint	Testing if the input for getAuthorDetail endpoint is a valid integer.	The third part API should be available and there should be at least 50 authors in our database.	The getAuthorDetail endpoint will be called with an id as a parameter.	The input id should be a valid integer.	Functional testing	Minor
Input check for getAuthorDetail endpoint	Testing if the input for getAuthorDetail endpoint is a valid id in our database.	The third part API should be available and there should be at least 50 authors in our database.	The getAuthorDetail endpoint will be called with an id as a parameter.	The input id should be a valid author id in our database.	Functional testing	Minor

Output check for getAuthorDetail endpoint	Testing if the output for getAuthorDetail endpoint will result in query error.	The third part API should be available and there should be at least 50 authors in our database.	The getAuthorDetail endpoint will call a controller which in turn run a query in our database.	The output of the query run in the database should not return an error.	Functional testing	Major
Search performance for authors	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 more than three characters in the database.	Chooses Author as the search type Enter the string to search. See the results.	The results should be shown within 2 seconds	Performance	Major
Search performance for papers	Test the performance of	Make sure there are at least 50 paper titles that	Choose Paper as the search type.	The results should be shown within 2 seconds	Performance	Major

	search for paper titles.	contain the same string which has more than three characters in the database.	Enter the string to search. See the results.			
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 five of them from the result Click Add button See that they are added to canvas with edges	The new nodes and their relations should be added to the canvas in 10 seconds.	Performance	Major
Layout	Test the performance of	Make sure there are 100 nodes on the canvas	Apply Cose-Bilkent layout	Every layout should be applied within 10 seconds	Performance	Major

	applying the layout		Apply Cola layout Apply Dagre layout Apply Euler layout Apply Klay layout			
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
Showing details of a Node	Test the performance of	Make sure there are 100 nodes on the canvas and at least one of time is	Click on an author node	After clicking on a node, details should be shown	Performance	Major

	showing details of a node	author and another one is paper	See the details of the node Click on a paper node See the details of the node	within 0.5 seconds		
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
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

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
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 edges are shown correctly in the canvas	The data provided by canvas should be correct	Reliability	Critical
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	Details and the clicked node should match	Reliability	Critical

			Check the shown details of the node are really the details of clicked node			
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
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	When clicking on different kinds of nodes, query options differ according to that	UI	Major

			Click on an author node Run the query options			
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, a alert should be shown When there are edges and nodes the canvas should be downloaded without any alert	UI	Minor
Upload	Test whether upload button works fine		When the canvas is empty, the upload button is clicked.	When the canvas empty, upload button should	UI	Minor

			When the canvas is not empty, the upload button is clicked.	work without an alert When the canvas is not empty, a confirmation message should be shown since the nodes in canvas will be lost		
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. Enter only three character and try	For the first two trials, the button should not be clickable. For the last trial, the button should be clickable	UI	Minor

			to click search button			
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	UI	Minor
Select All and Select Non Options	Test select all and select none buttons after a search	Make a search and have at least three results	Click select all button Click select none button Select at least one result	After every step the selected nodes should match the expected function of select all and select none button	UI	Major

			Click select all button Unselect at least one result Click select non button			
Clickable Filter Button according to Canvas	Test the clickability of the filter button	Canvas is empty.	Make a search and add a few nodes to the canvas. Choose at least one filter. Click on filter button Delete all nodes from the canvas	In the first trial, the button should be clickable and works fine. In the latter one the button should not be clickable	UI	Minor

			Click on filter button			
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. Click the filter button. Choose a new start time that is after end time.	Apart from the last trial, the filter button should work fine. For the last trial, an alert message should be shown to indicate start time cannot be after the end time.	UI	Major

			Click the filter button.			
Clickable Filter Buttons	Test the clickability of the filter button	There are nodes on the canvas	Not fill any filter area (time, journal name, etc.) Click the filter button Fill at least one of the filter area. Click on the filter button.	When no area is filled, an alert should be shown. When at least one area is filled, The filter button should work fine.	UI	Minor
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.	UI	Minor

7. 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 will allow 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 should be 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 will be a web application, and will not have a direct effect on the environment.

Economic: PaperAtlas does not directly affect the economical situation of its users.

Table 7: 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	-

8. 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 allows us to overcome many difficulties we had and will have 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 are not

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

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

8.1 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 is making great contributions in managing the necessary repositories and keeping our report web page up to date. Apart from management roles, Hakan is developing the responsive and user friendly frontend of our project along with Zehra. He is making 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 manages the database by writing scripts to batch process the data in our database. Currently, he is also contributing to the backend side of the project along with Aybala and Selbi.

Aybala Karakaya:

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

Selbi Ereshova

Selbi is making great contributions by keeping track of the upcoming reports and ensuring correct formalization and submission of our reports. She is 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.

Zehra Erdem:

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

8.2 Helping creating a collaborative and inclusive environment

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

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

To create an inclusive environment, we always support 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 is mindful of the team members' other commitments which help 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 arranges and runs the meetings we have regularly and that is a key factor in having collaboration. While running the meetings, she makes sure that everyone gets an opportunity to speak which helps us get an inclusive environment.

Selbi Ereshova:

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

Zehra Erdem:

She manages the project and checks on the work done. Since she also leads the process of work division, we are able to collaborate easily. When dividing the work, she makes sure that everyone gets an equal chance to work on impactful tasks. Therefore, she helps us obtain an inclusive environment.

8.3 Taking lead role and sharing leadership on the team

Team members share leadership on the team by taking on the lead role in different tasks. This is done in order to let every team member learn how to take the responsibility of a whole team. The tasks that every team member is 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

9. References

[1] “Google scholar.” [Online]. Available: https://scholar.google.com/. [Accessed: 10-Mar-2023].

[2] “Literature map software for Lit Reviews & Research,” *Litmaps*. [Online]. Available: https://www.litmaps.com/. [Accessed: 10-Mar-2023].