Delft University of Technology

CSE2000 Software Project

AIP++ (Extending AIP): Project Plan

Technical Writing Group **5C**

Bianca-Maria Cosma
Przemysław Kowalewski
Oskar Lorek
Michał Okoń
Jakub Tokarz

Project Coach | Dr.ir. Bart H.M. Gerritsen
Responsible Teaching Assistant | Dan Andreescu

Client | Alexandru Iosup, @ Large Research - `https://atlarge-research.com/`

Contents

1 Problem Analysis

Context. AIP is a project which facilitates parsing of article metadata, developed by the @Large Research team and currently available on GitHub [1]. One key feature of this tool, in its current form, is that it unifies three different data sources, namely DBLP, Semantic Scholar, and AMiner, and aims to overcome various data formatting issues. A user of the application can perform basic queries on the database with the GUI, or manually write more complex queries. It is also important to mention that the selection of data sources indicates that the system is specifically built for articles belonging to the field of computer systems [2].

Our task is to extend AIP, by adding features useful for researchers in this field, and updating the GUI accordingly: we call this extended application AIP++. The client offers us a great deal of flexibility for these additions, but also points out some basic features that are expected. Section 4 will formalize all of the requirements for the final version of our project. The following subsections give an introduction to the problem and serve as a basis on which we formulate these requirements.

1.1 Problem Description

First, we establish an overview of the problem and the main components we have identified, both from the project description, as well as our interviews with the client:

- **The potential for other features.** As explained previously, the current AIP version allows users to query the article database after parsing is completed. Our client suggested that new functionality is required to complement this and aid researchers using the tool. Among others, we intend to add support for the following features:
  - identifying *rising stars* [1] in the research community;
  - determining groups of authors that frequently write papers together;
  - detecting popular keywords relevant to the field of research (one might also refer to these as *hot* keywords);
  - allowing users to save queries and use them later;
  - exporting and importing.
- **Poor usability.** AIP is not easy to use, partly due to its simplistic GUI, which also lacks support for complex queries. This in itself is unfair to inexperienced users, who are not able to write queries manually. We also note that downloading the data source and having PostgreSQL installed on your machine are both prerequisites for running the application. To this end, we set three intentions for our project:
  - improving the aspect of the GUI;
  - extending the GUI to enable complex queries;
  - eliminating the need for hindering prerequisites, such as the ones previously mentioned.
- **The need for optimizing and extending the database.** As we are working with databases potentially containing millions of articles, and for the sake of maintainability, we also focus our attention on improvements related to query execution time and changing the current database schema:
  - indexing the database;

[1] When referring to *rising stars*, we are targeting new academics whose research has a significant impact on the field. We consider a researcher's academic age, as well as the quality of their publications.

- adding a graph database for better performance on certain queries;
- implementing parallelism;
- splitting the abstract to allow for easier keyword detection;
- parsing an additional data source.

- **Lack of visual feedback.** There are cases when a user should be able to visualize the results of a query in a way that is more intuitive than a list. We aim to add as many visualizations as possible, especially for those features that can be intuitively mapped to graph representations, such as identifying recurrent co-authors or article citations.

1.2 Overview of Project Objectives

We aim to build an application tailored to academics, with an intuitive interface and good performance. We therefore establish the following goals for our project:

- **Build a functioning application that meets at least *Must Have* and *Should Have* requirements.** We acknowledge that problems may arise during the course of the project, and so we might not be able to meet all of the requirements outlined in section 4. However, we aim to at least cover those features which are essential to our client.
- **Deliver a product that is maintainable, documented, and thoroughly tested.** We believe the right thing to do is to focus on quality, rather than quantity. As this is an open-source project, it is most important that other developers can build upon our code and improve the application further.
- **Write a final report which documents our research and the project's functionality.** We wish to focus on several research questions, including, but not limited to, the following:
  - algorithms used to identify *rising stars*;
  - performance comparisons between graph and relational databases;
  - compelling data visualisations of bibliometric networks;
  - methods to identify *hot* keywords;
  - finding authors that often write papers together.

1.3 Stakeholders

A product stakeholder can be defined as a person influenced by our product. This influence can be either positive or negative. In other words, a stakeholder either benefits or loses something as a consequence of the project [3]. Identification of the stakeholders is a non-trivial task requiring us to look at the whole structure of our project from a wider perspective. A thorough analysis of all the factors impacting and being impacted by our work not only allows us to develop a functional application, but also ensures it is correct from an ethical standpoint.

We have managed to identify the following stakeholders:

- **Authors of articles in the databases.** The databases that are already incorporated into AIP, as well as the ones that are still to be added, contain names of researchers who have devoted their valuable time to their work. The way our application works could potentially discriminate against some of them. We should give special attention to ensuring that all of their work is uniformly accessible to the users reaching the databases through our interface.

- **Owners of the data sets used.** Semantic Scholar, DBLP and AMiner are the entities currently involved in providing our application with data sets. These data sets contain research information used to supplement the main database. We are responsible for handling these archives appropriately and, therefore, we should strongly avoid modifying the information contained in them. As we add more data sources to our project, we will have to apply a similar approach to new data owners.

- **Application users.** Undoubtedly, most of the potential users of our application are people searching for scientific articles in order to conduct research. The validity of the research can be highly influenced by the kind of results they find. Moreover, some of the features may contribute to rising popularity of particular scientific fields and scientists.

- **Our client - @Large Research.** @Large Research is a research group specializing in Massivizing Computer Systems. Our client expects us to develop an application on top of the components they have already created. They demand the application to be scalable and maintainable, such that it can be expanded in the future. We communicate with our client on a regular basis to control whether our results meet their expectations.

- **Our teacher mentor and teaching assistant.** These are the people responsible for assessing the process of the application development, as well as grading the final product. Their feedback can possibly influence our workflow and introduce changes to the product.

- **Delft University of Technology.** Since the project is a part of the Computer Science and Engineering curriculum at Delft University of Technology, our work is bound to represent the name of the university itself.

- **We - the developers.** Finally, we, as the developers, are influenced by the outcome of our work as the entities directly involved in the development process and responsible for the delivery of a functioning product.

1.4 Competitor Analysis

Before delving into the content of this section, it is noteworthy to specify that we are researching similar products (or, as we call them, competitors) simply to determine the standards that an application of this type is expected to meet. There is no monetary gain involved, so we regard everything from a research perspective, rather than a business-oriented point of view, and **we restrict our analysis to products that are free to use.** For this reason, some relevant similar products are not included in this study. For instance, Elsevier's *Scopus* [1] is a widely used academic search engine with great performance, which produces reliable query results and provides many advanced search options. Unfortunately, it is also paywalled.

It is important that we do not work on a tool that has the exact same features and performance as one that is already on the market. Additionally, we would like to ensure that our efforts are aimed at creating a product that is useful and has the potential to become successful.

1.4.1 Identifying Similar Products

Our first step is to single out some products which are similar to our application, and establish a framework to quantify this similarity. To this end, we conduct a short study based on *market commonality*, "the degree to which they [competitors] address similar customer needs" and *resource similarity*, "the degree to which their resource endowment is similar in terms of type or

[1] `https://www.scopus.com/`

composition" [4, p. 160]. Figure 1 shows the products we have identified, and also highlights the categories to which they belong.

Note how resource similarity is not defined in terms of amount, but in terms of type. We acknowledge that most our competitors have an amount of resources that far exceeds our own, at least in terms of manpower.

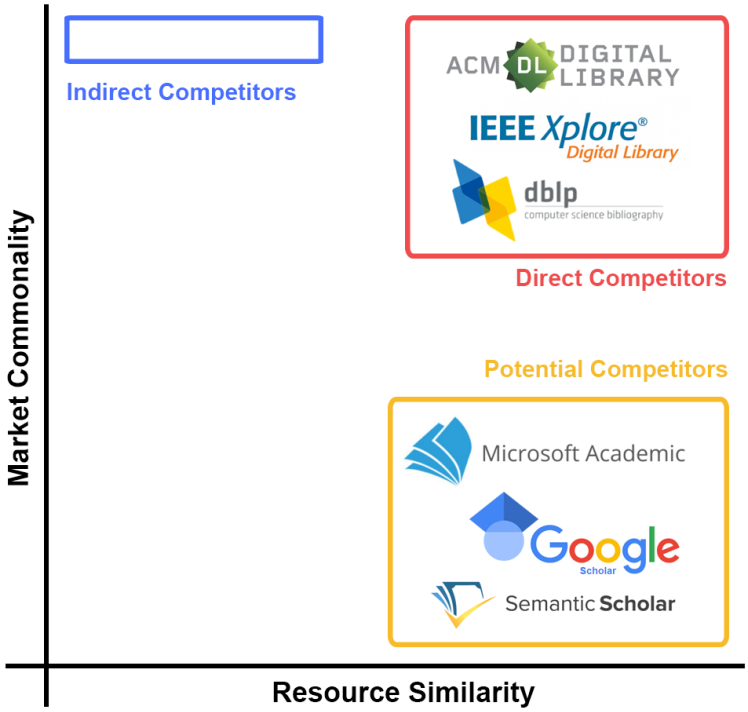

Figure 1: Competitor analysis for AIP++, as seen in [4, fig. 1].

We will move on to a brief discussion about the similar products we have identified, focusing on their best features, as well as what functionality they lack. Our intention is to see how popular academic search engines implement certain functionality, to get inspired by their best features, and to determine what contributions we can bring to this market.

We have not yet identified any products that could be classified as *indirect competitors*. Therefore, we are going to focus on the other two categories.

1.4.2 Potential competitors

Potential competitors have resources that are similar to ours, that is, databases containing information about scientific articles, journals, and researchers. However, we consider that these products are different in terms of market commonality, because they are not specifically designed for researchers in the field of computer systems, but for the research community as a whole.

Google Scholar [2] is perhaps the most well-known search engine for any type of scientific publication, and it also includes author profiles. It is straightforward to look for a specific author or a set of words appearing in the title of an article, although some advanced search options are

[2] `https://scholar.google.com/`

available as well. *Google Scholar* also makes it possible for users to create alerts for a certain query, and to be notified via email if more results become available. One drawback we want to point out is that this search engine does not allow distinction between different fields of research, which in many cases is an essential feature.

In terms of layout and querying, *Semantic Scholar* [3] is quite similar to *Google Scholar*, but has some distinguishing new features, such as easy filtering based on the field of study or the publication type. In addition, it is possible to receive updates about a certain topic, and this is the case for *Microsoft Academic* [4] as well. Aside from providing a lot of the functionality implemented by *Semantic Scholar*, Microsoft's academic search engine has support for browsing related topics and shows a wise variety of statistics and visualizations, some examples being author-journal and inter-topic relationships.

1.4.3 Direct competitors

We regard direct competitors as ranking high in both market commonality and resource similarity. These are products created for the computer systems research community.

The advanced search in *IEEE Xplore* [5] allows for relatively flexible querying, and many filtering options are supported, including the choice of topic, which could be something like *data mining* or *neural nets*. The homepage displays a list of featured articles and authors to whom a user can subscribe, although the criteria on which they are selected is unclear. The results of a query and the history of queries can be exported, and functionality is available to search within the results of a query.

In comparison, *ACM Digital Library* [6] does not have any distinguishing features, and its functionality and article database are limited. Lastly, *DBLP* [7] can be best described as a bibliography, rather than an academic search engine, but it is still a valuable resource for the community.

1.4.4 The Role of AIP++ in the Research Community

We note that two of the products we have mentioned, DBLP and Semantic Scholar, provide data sources that are used as the basis of our application. Therefore, it is unnatural to think of them as competitors, but we still used this term for the sake of conducting a proper analysis.

We believe that AIP++ can have a positive impact on the research community, most of all because it will be open-source and accessible to all those who wish to make a contribution. Aside from features inspired by popular academic search engines, we want to include engaging data visualizations of the computer systems research network, and to give users the freedom to write custom SQL queries. We believe that the interfaces of many of the previously mentioned products limit the user's options for querying the article database, and one of our goals is to offer an alternative that allows for more freedom and creativity.

Our client is part of a research team (@Large Research), and, among others, has expressed interest in a *rising stars* feature, promoting researchers who are quickly becoming successful, and whose recent work in a field can be described as innovative and outstanding. This can be very valuable for senior researchers and leaders in a community, because they get the opportunity to contact these young pioneers and offer them junior positions, in academia or in the industry [5]. We expect that laying the foundations for such a feature can be one of the main highlights of AIP++. Additionally, AIP++ users will have the option to save queries and reuse them later, and to view authors that frequently write papers together.

[3] `https://www.semanticscholar.org/`
[4] `https://academic.microsoft.com/home`
[5] `https://ieeexplore.ieee.org`
[6] `https://dl.acm.org/`
[7] `https://dblp.uni-trier.de/`

2 Feasibility Study

Ian Sommerville defines the feasibility study as an important part of the requirements engineering process [6]. Such a study is conducted to determine if the project goals are attainable given certain constraints, and to give the development team an indication of whether or not they should proceed with the project or rethink their strategy. We believe this is an essential part of our project plan. Therefore, the focus of this section is to determine whether it is feasible to complete this project under the technical, resource and time constrains that we have to deal with.

2.1 Technologies and Frameworks

From the technical feasibility point of view, we believe that the level of technology available in the software development space is sufficient to develop the required product. A basic version of the parser for our application already exists. It is our task to improve upon it and to create a fully functioning web application.

Undoubtedly, software and data engineering are the most vigorously expanding branches of computer science nowadays. The market is overwhelmed by the surge of new applications that are released daily. This goes hand in hand with a multitude of available frameworks aiming to simplify the development of such applications, including ours. Therefore, various technologies can be applied to each layer of the software. We will explore the applicable possibilities further in the Section 5.

For the improvement part, which concerns the performance of the database, we will make use of a graph database (Neo4j) and an ORM technology for which plenty of frameworks are available. In order to create the back-end of the web application, there are also more than enough frameworks which are suitable for AIP++. They are open-source and offer a wide range of features which greatly simplify the development process for the server side of the application [7]. See section 5.2 for a more detailed discussion.

2.2 Time and Resource Constraints

Important factors affecting the feasibility are the time and resource constrains imposed by the duration of this course. As the developing party, we created a prioritised list of requirements with the approval of the client, which is included in section 4 of this report. These requirements adhere to the MoSCoW method, with *Must Have* requirements being our top priority during the development process. With five team members and each member of our team being expected to work approximately 38 hours a week [8] for the duration of this quarter (10 weeks). We believe that we will have enough time and manpower to at least meet the basic requirements. In order to visualize our timeline and to realistically distribute the workload across these 10 weeks, we created a Gantt chart, which can be found in appendix A.

2.3 Potential Changes

In the unfortunate, yet unlikely case in which the project would turn out to be infeasible, there are a few changes which we could propose to the client. These changes would consist of dropping some of the *Could Have* requirements or proposing simpler, yet less efficient, solutions to the agreed issues. An example of such a change would be putting less focus on the import/export functionality or potentially not implementing the functionality needed for storing *favorite* queries. Although such features would be a great addition to our product, they did not receive such high priority (they belong to the *Should Have* category).

[8] In order to obtain 15 EC, it is estimated that a student should work 42 hours a week. From this, we subtract 4 hours, since this course also consists of other modules, such as Technical Writing.

2.4 A Verdict on Feasibility

To summarize our feasibility study, we believe that under the given time and resource constraints, as well as the available technologies and frameworks in the software development space, the development of our product with our client's requirements is feasible.

Finally, given the analysis outlined in this section, as well as the problem description, we highlight the project's feasibility by answering the following three questions [6, p. 100]:

- **"Does the system contribute to the overall objectives of the organisation?"** Yes. We are developing a free, open-source project, which aims to help researchers in the field of computer systems. This is is line with the values of the @Large Research team.
- **"Can the system be implemented within schedule and budget using current technology?"** Yes. With the time and manpower resources at hand, as well as our MoSCoW prioritisation strategy, we believe that we can develop a functioning product before the deadline. A more thorough analysis of available technologies and frameworks is included in our solution proposal (section 5).
- **"Can the system be integrated with other systems that are used?"** Yes. During the development process, the current parser will be incorporated into the larger application we will build. The final product is a stand-alone application which does not yet require integration with any other systems.

3 Risk Analysis

Every project may encounter issues impeding its successful completion. We decided to conduct a risk analysis in order to identify, categorize and estimate the likelihood of these events happening, as well as their severity. Having done so, we developed a concrete risk management plan that minimizes the probability of hampering events taking effect and lessens the negative impact following the occurrence of such issues. Moreover, we created a risk monitoring process, which evaluates the likelihood and potential severity of these mishaps, allowing us to act accordingly.

By conducting the above-mentioned, complete risk analysis, we maximize the chances of successful completion of the project to the desired standard.

3.1 Hazard Identification Process

Firstly, we decided to identify the issues that can arise throughout the project. We were particularly interested in the events that are highly probable to occur and that would have severely negative consequences for the result of the project. Those are the issues that we have to analyze further.

We decided to commence our investigation by conducting an interview with the client. Since the client has great experience with mentoring group projects (including CSE2000), we considered their insight very helpful when identifying potential issues. We were particularly interested in the kind of problems that previous Software Project groups encountered and how they managed to handle them.

After the interview with the client, we held several group meetings in which all members were required to list potential issues that can arise during the project. We listed the issues individually, so that we were not biased by other members' opinions and thus more potential points of concern could be identified.

As a next step, we combined the lists together. Each member of the group was required to rank the probability of potential hazards (1 – nearly impossible, 10 – almost certain), and the effect that it would have on the outcome of the project (1 – nearly no visible consequences, 10 –

disastrous consequences). Events that achieved an average probability score of 2.0 or below (for example: "GitLab repository lost") or that achieved an average outcome score of 2.0 or below (for example: "Group member being late to the meeting") were excluded from further risk analysis. The higher the average combined score of the event was, the more attention was devoted to it during the development of the risk management plan.

3.2 Identified Risks

After conducting the hazard identification, we analyzed the following risks:

- **The product is used for hiring researchers.**

  One of the product features is the *rising stars* functionality. Thanks to it, the decision makers may select potential candidates, judging by how our tool ranks them. This can result in a negative bias. In consequence, the deployment of our product is connected with ethical issues, since it may lead to rejection of qualified candidates. Therefore, the consequences of this hazard could be "very harmful". Moreover, the probability of this hazard happening is positively correlated with the popularity of our application, thus it is hard to estimate its likelihood. At this point, we think we can rank it as "probable".

- **One or several group members get overwhelmed by the amount of work.**

  Although we put a lot of effort into splitting the tasks fairly, such that the estimated amount of work for each member is equal, it may happen that some group members are overwhelmed by the workload during certain weeks. This can be due to underestimation or overestimation of work required to complete certain tasks. We estimate the probability of this event as "highly probable", due to the fact that we are not familiar with all of the tools used through the project, like Django, as a framework, or Python, as a programming language. Consequently, we are unable to precisely measure how much time each task consumes. We classified the severity of consequences of this event as "very harmful for the outcome", as it can lead to one of the *Must Have* requirements not being completed.

- **One or several group members loses enthusiasm.**

  Loss of enthusiasm by one of the team members, resulting in under performance, is a possibility that cannot be completely dismissed. We estimate the probability of this event as "probable". During the pandemic, we live in very static environments, which can have a negative impact on our well-being and overall productivity. We feel that the consequences of this would be "moderately harmful for the outcome", as we would need to reevaluate our priorities and resign from implementing some of the *Could Have* and *Should Have* requirements.

- **Group member drops out of the course.**

  Requirements to enroll for the course CSE2000 are very strict and demanding, and thus only highly motivated students attend the course. The consequences of not completing the course are severe, as they are likely to result in a one year study delay. Therefore, we feel that it is "unlikely" that one of the group members will drop out of the course. Nevertheless, such possibility cannot be excluded due to personal circumstances. We estimate the potential consequences of this event to be "very harmful for the outcome" as it can lead to not finalizing one of the "Must Have" requirements.

- **We do not have enough time to complete the report at desired quality.**

  During the interview, the client mentioned that last year's group underestimated the time required to complete the code. Subsequently, they did not have enough time to hone the report. Thus, we estimate the probability of this event as "highly probable", especially given

the preferable length of the final document (15000 – 20000 words). We estimate that low quality of the report, caused by insufficient time, would be "harmful for the outcome" as we would not be able to effectively show our work, nor suggest possible further improvements to the product.

3.3 Risk Management

Having performed the risk analysis, our group decided to adapt the process to minimize the likelihood of the hazards and reduce potential negative impact if those hazards happen.

We decided to incorporate the following in our process:

- To minimize impact of one of the group members dropping out of the course, we will practice pair programming, meaning that for each part of the code base, there are at least two people responsible for it.
- To minimize the impact of one of the group members getting overwhelmed by the amount of work, we will follow an agile development process, such that if one of the group members feels that (s)he is getting overwhelmed, his/her tasks can be flexibly redistributed to other team members.
- To reduce the probability of members losing enthusiasm throughout the project, we will conduct 3 meetings per week for the purpose of socializing and project updates.
- To reduce the probability of not having enough time for writing the report, we are aiming to finish the product by the middle of week 9.
- To minimize the negative impact of our product being used for hiring researchers, we decided to provide very transparent documentation for the *rising stars* algorithm we are going to implement. Moreover, we decided to include a warning within the *rising stars* tab, stating that if someone wants to use this feature as a performance measurement, (s)he should read the documentation to understand how the algorithm works. By increasing the transparency of the algorithm, we would make the hiring manager more aware of the upsides and downsides of the *rising stars* feature.

3.4 Risk Monitoring

Throughout the project our estimates of the probability and potential consequences of hazards are prone to be changing. Thus, our group has agreed to participate in risk monitoring process. During each sprint review, we will reevaluate our estimations of hazards and change our risk management plan accordingly to adapt to possibly emerging risks.

4 Requirements

Elicitation. After interviewing our client several times during the past few weeks, we formed a good understanding of the requirements our project would need to satisfy. In order to gain more insight into the problem, we decided to interview one of our stakeholders, in particular, someone who is part of the research community and frequently has to search for scientific articles. Our interviewee, Prof. L. Barbu, Faculty of Mathematics and Computer Science, at *Ovidius* University, Romania, was not proficient in SQL, so we also had the chance to focus on the visual querying capabilities of our application. The full set of questions and answers can be found in appendix B, where we also draw some of our own conclusions regarding the interview.

4.1 Functional Requirements

Functional requirements describe the features of the application, as well as its behavior and reactions to user inputs [6]. We have decided to prioritize functional requirements using the MoSCoW method, which is common for teams that use the agile approach to software development. Under time pressure and unforeseeable obstacles, it is our intention to deliver a product that can still be useful to our client. Functional requirements can be distinguished into four categories [8]:

- ***M**ust Have* - requirements that have to be implemented for the final product to be accepted by our client;
- ***S**hould Have* - requirements that are almost as important as the previous category, but can be overlooked if the team faces an extreme challenge;
- ***C**ould Have* - requirements that describe features the client would like to have, but does not prioritize as much as the first two categories;
- ***W**on't Have* - requirements that can be added to the application in the future, but will not be implemented by our team during this run of the project.

The following subsections show our requirement prioritisation strategy, which was formulated in consultation with our client. We note that, when referring to articles, we are only considering those from the field of computer systems.

4.1.1 *Must Have* Requirements

1. The user can browse the database using the GUI.
2. The user can filter based on columns which have a numerical data type, by specifying a range.
3. The user can search for a specific value in each of the database columns.
4. The user can see all of the available information about a paper in the database, that is: the id, the title, the authors, the publication venue, the volume of the publishing journal (only for some papers), the DOI, the abstract, and the number of citations.
5. The user can see all of the available information about an author in the database, that is: the id, the name, and the ORCID, the latter only if available.
6. The user can sort the database table lexicographically.
7. The user can execute custom SQL queries on the database, by typing them out and running them.
8. The user can only query and view the data, not modify it.
9. The user can export the results of a query, as well as the query itself. The exported file will contain the version of the database on which it was executed (that is, information regarding when the database and data sources were last updated).
10. The database is automatically and periodically updated, and the user is informed about it.
11. The user can see which versions of the data sources are being used, as well as when the database was last modified.

4.1.2 *Should Have* Requirements

1. The user can click on the DOI of a paper to be redirected to the web-page where the publisher posted that paper.
2. The user can access data from the Microsoft Academic Graph database, through our application.
3. The user can save a query, marking it as *favorite*.
4. The user can see the list of his/her favorite queries.
5. The user can execute a *favorite* query by clicking on it.
6. The user can remove a query from his/her list of favorites.
7. The user can export his/her list of favorite queries.
8. The user can import a list of favorite queries.
9. The user can see the SQL query that was executed after every search.
10. The user can see who are the *rising stars* for articles containing a given keyword.

4.1.3 *Could Have* Requirements

1. For each author, the user can see other authors with whom (s)he has collaborated.
2. For the previous requirement, the user can also see a graph visualization, meaning that (s)he can see if there is a strong connection between certain groups of authors.
3. The user can see what the trending topics are (we call these *hot* keywords).
4. Given the resulting list of *hot* keywords, the user can click on one of them and see which articles it is linked to.
5. For each article, the user can see a list of articles which cited it.
6. For each article, the user can see which articles it cites.
7. The user can give a name to each *favorite* query.
8. The user can search for a *favorite* query in his/her list, by providing its name.
9. The user can click on an author to reveal more information about him/her.

4.1.4 *Won't Have* Requirements

1. The user has access to articles from fields other than computer systems.
2. The user can see a history of queries.
3. The user receives recommendations regarding papers that might interest him/her, based on his/her queries.

4.2 Non-Functional Requirements

To show how the system as a whole is defined by certain constraints imposed by our client, we specify the non-functional requirements that it should fulfill. Among others, these often describe the performance of the application in terms of speed or memory, the programming language that has to be used, the operating system on which the application should run, or security standards.

Non-functional requirements can be broken down into three categories [6]: *product* requirements, *organizational* requirements and *external* requirements. We mention that our client was not very strict with regard to these requirements, and we list those that we have identified:

1. The system will be open-source. The source code of the application will be published after the project is completed. (*organizational*)
2. The final version of the source code will be available on the @Large Research GitHub page. (*organizational*)
3. The product will be developed as a web application. (*organizational*)
4. The system will be protected against malicious users. The database will be available as read-only, and no queries should modify it. (*product*)
5. The latency of the application will be in the order of seconds, that is, it should never exceed one minute. (*product*)
6. As compared to the current version of AIP, the performance of the database will show improvement in terms of latency, due to the use of indexing and other suitable techniques. (*product*)

5 Solution Proposal

Our solution proposal consists of the requirements listed in the previous section, as well as the general architecture, which includes specific frameworks that will be used. We also considered the future prospects of AIP++, placing emphasis on creating an easily maintainable code base.

We are aiming to make our product as flexible as possible by allowing the users to choose whether they want to host the application locally or access it remotely through the web browser.

5.1 Architecture

Looking deeper into this project, we realized that the application can be easily divided into three layers. These layers perfectly align with the Model-View-Controller (MVC) architecture, with the front-end as the view, back-end as the controller and database as the model. MVC is an example of a Layered Architectural Pattern, which represents one of the most widely used approaches nowadays [9, p. 80]. Following it should satisfy our project's need for simultaneous development, ease of modification and frictionless testability.

Moreover, this division directly leads to a smooth distribution of work among the group members, considering that every layer acts as a separate entity. As a result, each component can be developed by specific teams, independently of one another.

To implement the MVC architecture, a typical use case of our application would look as follows:

- The controller receives an HTTP request sent by the user.
- The controller contacts the model to retrieve the requested data.
- The controller performs appropriate operations on the data.

- The results of the data manipulations are sent to the view layer.
- The view layer renders a display for the user to see.

5.2 The Back-End

5.2.1 A Comparison Between Frameworks

There are various open-source frameworks available, offering a range of features which aim to simplify the development process for the server side of the application [7]. In this section, we are going to mention and summarize the most popular ones.

- Ruby on Rails - Ruby-based open-source framework designed for web applications. Rails support multiple pre-made modules, reducing the time needed for the construction of the most generic code. Moreover, the framework provides special tools designed to facilitate testing. We have rejected this framework, as we have no knowledge about Ruby and learning its basics could waste too much of the resources at hand.
- **Spring Boot** - framework well-known to TU Delft students. Spring Boot helps set up Java web applications and provides programmers with more advanced functionalities, which we will most likely not need. Besides, the already existing database code is written in Python. We abstain from using many programming languages in order to reduce the complexity of our project and to avoid the need to integrate components written in two different languages.
- **Flask** - simple and flexible Python-based framework that does not require components from external sources to function. Unfortunately, this framework does not support a database abstraction layer, making communication with the databases difficult to implement.
- **Django** - Python-based framework combining all the favorable aspects of the frameworks mentioned before. Its gentle learning curve and support for various web appliances should greatly accelerate the development process and allow us to create an application with every core feature mentioned by our client. The key characteristics of this framework are explored further in the next section.

5.2.2 Django

To facilitate the development process, we looked at the most popular back-end frameworks. Undoubtedly, all of them offer range of functionalities that we could find helpful. After careful research, we have opted for a solution using REST API along with the Django framework - one of the most popular frameworks used by websites such as Instagram and Pinterest. There are plenty of reasons for this choice.

A familiar programming language. Django is a framework for Python. The existing code, which we are expected to expand with new functionalities, is written in Python. That makes the coding process more seamless, as we do not have to overcome issues connected with combining multiple programming languages in one project. Moreover, Python is an easy-to-use language, which we have already used throughout the course of our bachelor's degree, therefore, we we will not incur any time overhead unavoidable when learning a new language.

Scalability, maintainability, and reusability. Django meets some of the most important requirements of our project - it is both scalable and maintainable. In the case of Django, back-end is divided into so-called applications. Each application is treated as a separate entity, which can be easily modified, replaced or removed without any interference with other parts of the code. Furthermore, new components can be freely added as the project grows bigger. That guarantees

scalability. Similarly, code reusability is highly encouraged by the framework, as the applications can be copied and reused in other programs.

A framework that can be easily integrated with the rest of the project. Lastly, Django is one of the most versatile frameworks available, providing tools to interact with databases easily and work with any user-side frameworks, making back-end and front-end completely independent in terms of the framework choice.

5.3 The Front-End

5.3.1 UI Approach

The major choice regarding the front-end of our application was choosing between a standalone desktop application and a web-based application in the browser. We went with the second approach, as it seemed to be more flexible. It could be run both locally or remotely, if someone wishes to host it. The second choice we had to make is choosing a component-based UI library. It is a very important part of the process, as making the right choice will accelerate the UI development process greatly by decreasing the amount of effort put into establishing the basic logic of our application. There are three major competitors on this market - **React**, **Angular** and **Vue.js**. We chose React for the reasons below.

5.3.2 Why React?

- It has been developed by Facebook and it is the most popular framework among the ones considered [10]. That means it is the most widely tested and has the greatest support from the community. It also possesses a large developer toolset.
- It is free and open-source.
- Our team members are most familiar with it.
- It is very flexible, which means that, if we decide to add a new functionality during the development process, it will be easily incorporable into the existing application.

5.4 The Database

5.4.1 Technologies and Libraries

Integration With Django. Other than being used as the backbone of the application, Django also includes default ORM. We will use it to directly communicate with the database from the controller layer. This greatly improves the maintainability of the project, allowing for complex business logic while still having effective and efficient queries.

PostgreSQL. PostgreSQL is used for the current version of the system. It makes no sense changing the RDBMS to a different one, since, primarily, PostgreSQL is among the most widely used ones [11]. Thus, its SQL dialect should be familiar for the majority of stakeholders that already know SQL. Moreover, the use of a relational database, in comparison to a no-SQL one like MongoDB, is desired, because the system is highly unlikely to become distributed, thus we do not have to trade the *consistency* guaranteed by an RDBMS for the *partition tolerance* offered by no-SQL systems.

Neo4j. Neo4j is the most popular graph database [12]. It is very popular in the industry and its libraries can be used to visualize the results of the queries. During the interview with the client, we discovered that currently the database is often queried recursively (think of queries specific

to networks, such as finding cliques). Graph databases perform significantly better for this type of queries, in comparison with relational databases [13]. Thus, we want to investigate possible performance gains, while trying Neo4j as an alternative.

Psycopg2. Psycopg2 is the most popular PostgreSQL adapter for Python. It is already being used in the base parser we have to extend. A major advantage of this technology is the thread-safety it guarantees. This aspect will be crucial when introducing parallelism to the application.

6 Software Development Methodology

6.1 Development Approach

We chose an agile software development methodology. We will have short Scrum meetings on Wednesdays and Fridays, during which each team member will share their progress and explain what they will be working on that day and the next two days. On each Monday, we will have a **Sprint Review** session and **Sprint Planning** for the next week. During each *Sprint Planning*, each team member will get assigned issues to resolve and new functionality to implement for the next week. We will maintain our **Sprint Backlog** in using GitLab's issue functionality, and use milestones to represent sprints.

6.2 New Functionality

There is a possibility that during the development we will find it necessary to include new functionality we did not realize was needed during the initial planning. To not obstruct the ongoing process of developing the core requirements, we came up with a special procedure. A specific plan for including new functionality to the current project can be presented at any team meeting by a team member able to motivate the need for these new requirements and prove that including them in our project plan will not disrupt the current workflow. The proposition will then be subject to a vote and, if the majority of team members agree with the proposal, the functionality will be added as the requirement.

We also consider the case in which the client is actually the one introducing new requirements later on the project, which is not uncommon for agile development teams. For more details, see appendix A.

6.3 Definition of "Done"

The definition of "done" is susceptible to interpretation. That is why we have decided to have a special process to consider things as done for features and tasks.

When a team member states during a team meeting that (s)he has finished a specific task and (s)he wants to merge their change to the *dev* branch, two steps need to be taken. First of all, the tests written for that component need to reach at least 80% line coverage, such that we fully incorporate testing into our development process. Secondly, the merge request has to be reviewed and then approved by at least two other team members. The reviewers are supposed to check if the implemented functionality works as intended and the new code is readable, well-documented and tested.

Lastly, we mention that we will create an issue template that will divide each task into multiple sub-tasks. If all of the sub-tasks are completed and the two previously mentioned requirements are met, we know that a feature is "done".

Bibliography

[1] "Github - atlarge-research/aip: An instrument to combine, unify, and correct (scientific) article meta-data," https://github.com/atlarge-research/AIP, (Accessed on 04/22/2021).

[2] L. Versluis and A. Iosup, "A survey and annotated bibliography of workflow scheduling in computing infrastructures: Community, keyword, and article reviews – extended technical report," 2020, (Accessed on 04/22/2021). [Online]. Available: https://export.arxiv.org/abs/2004.10077v1

[3] I. Alexander and S. Robertson, "Understanding project sociology by modeling stakeholders," *Software, IEEE*, vol. 21, pp. 23 – 27, 02 2004.

[4] M. Bergen and M. Peteraf, "Competitor identification and competitor analysis: A broad-based managerial approach," *Managerial and Decision Economics*, vol. 23, pp. 157–169, 06 2002.

[5] A. Daud, M. Song, M. K. Hayat, T. Amjad, R. Abbasi, H. Dawood, and A. Ghani, "Finding rising stars in bibliometric networks: a survey," *Scientometrics*, vol. 124, pp. 633–661, 04 2020.

[6] I. Sommerville, *Software Engineering*, 9th ed. USA: Addison-Wesley Publishing Company, 2010.

[7] J. Clark, "Top 10 backend frameworks," https://blog.back4app.com/backend-frameworks (Accessed on 04/22/2021).

[8] A. Cline, *Agile Development in the Real World*. Apress, 2015.

[9] M. Keeling, *Design It!: From Programmer to Software Architect*. Pragmatic Bookshelf, 2017.

[10] W3Techs, "Market share trends for javascript libraries," https://w3techs.com/technologies/history_overview/javascript_library (Accessed on 04/22/2021).

[11] S. Juba and A. Volkov, *Learning PostgreSQL 11: a beginner's guide to building high-performance PostgreSQL database solutions*. Packt Publishing Ltd, 2019.

[12] D. Fernandes and J. Bernardino, "Graph databases comparison: Allegrograph, arangodb, infinitegraph, neo4j, and orientdb." in *DATA*, 2018, pp. 373–380.

[13] S. Batra and C. Tyagi, "Comparative analysis of relational and graph databases," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 2, pp. 509–512, 2012.

A Project Schedule

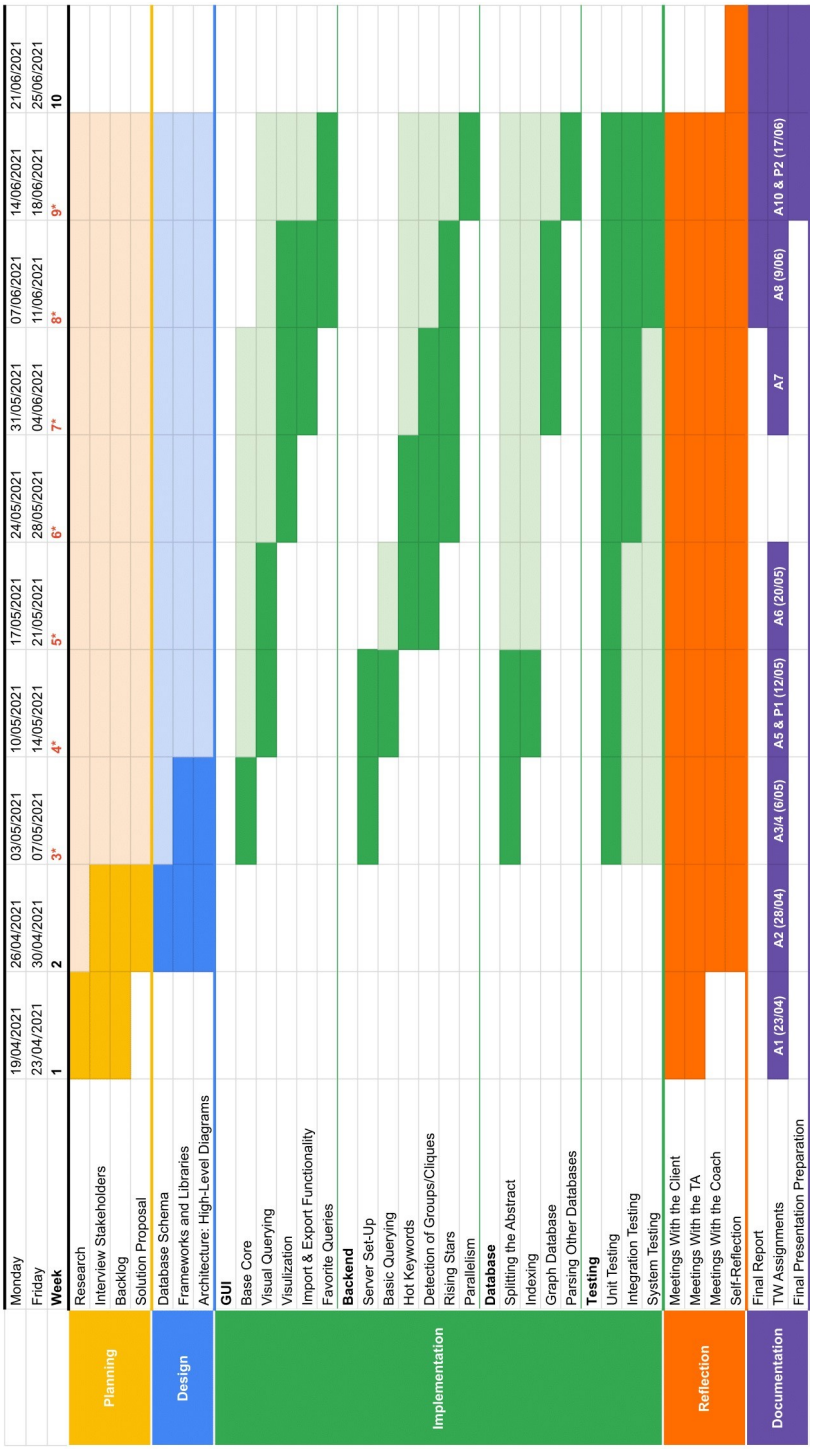

Figure 2: Project schedule for AIP++.

Figure 2 shows a Gantt chart containing the time estimates we have identified for each stage in our project. We acknowledge that working with this kind of chart is not common practice for teams that follow an agile approach. The main purpose of this schedule is for us to have reference to the ideal outcome of this project.

We expect that there are still dependencies we have not identified, and implementation efforts that we have underestimated. Moreover, we need to easily integrate new requirements that might be added by our client during the course of the project. Lighter colors are therefore used to indicate that the schedule is subject to change, making this Gantt chart more *agile-friendly*. Our sprints begin on Mondays, for each of the weeks marked with an asterisk.

B Requirements Elicitation

B.1 Interview With One of Our Stakeholders

- **What platforms are you currently using to search for scientific papers? Are the platforms free?**

  Answer: Mostly *ResearchGate* and *Google Scholar*, which are both free, as well as *Web of Science*, which is not free.

- **Does lack of SQL proficiency hinder you from effectively searching for relevant scientific papers?**

  Answer: I have heard of SQL, but I do not see how it can be relevant when searching for scientific publications, or how it can make things easier for me. I never had to use it, and all search engines I used did not require me to do so.

- **Would filtering options such as explicitly specifying the name of the author, year of publication or keyword search, satisfy your querying requirements?**

  Answer: Most of the time, I am interested in the title of the article or the author. However, sometimes I would like to search for articles in a specific journal, and it is also useful, for a specific article, to see what other articles included it in their references.

- **Would you be interested in a *rising stars* feature, revealing recently often cited researchers?**

  Answer: Yes. Many of these researchers are potential "invited speakers" for a conference, for instance, but there are other reasons you might want to get in touch with them. However, most journals often show you which recent articles were cited the most. I am not sure how a list of "rising stars" would be different, but I think it could be interesting.

- **Would you be interested in a *hot* keywords feature, showing the articles concerning the trending topics in your field of research?**

  Answer: That is not a feature that would help me, in particular. Usually, if you work within a field, you are not interested in trending topics, because it is very difficult to change your field. So, if a topic is trending, that does not affect me much. I can see how this can be useful for new researchers, as it might be helpful to start your career focusing on a trending topic.

- **Are there any other features that would help you effectively search for scientific papers?**

  Answer: For an author, I would like to see a list of journals where s(he) published articles, and, for each journal, how many articles (s)he published there. It is a bonus if you can also

see how many times an author was cited from articles published in a certain journal. This can be helpful in many ways.

B.2 Findings

Our interviewee stated that she uses *Google Scholar*, which is one of the products we have mentioned in our competitor analysis (section 1.4). She indicated that she does not use SQL, because she does not need to: the alternative of searching by filling out specific fields is satisfactory. This correlates with the requirements given by our client, as well as our research, where we came to conclusion that a user should not be required to know SQL to efficiently search for scientific papers. Other features that our client requested, such as *rising stars* or *hot* keywords were also met with approval from the interviewee. She expressed curiosity regarding the *rising stars* functionality, and mentioned that trending topics are a feature that she would not use personally, but understands how her colleagues would find it useful.

Acronyms

AIP Article Information Parser. 1, 3, 4, 6–8, 14, 19

API Application Programming Interface. 15

DOI Digital Object Identifier. 12, 13

GUI Graphical User Interface. 3, 12

HTTP HyperText Transfer Protocol. 14

MVC Model-View-Controller. 14

ORCID Open Researcher and Contributor ID. 12

ORM Object-Relational Mapping. 16

RDBMS Relational Database Management System. 16

REST REpresentational State Transfer. 15