



Cairo University
Faculty of Computers and Artificial intelligence
Department of Networks and Cybersecurity



Xtract

Supervised by
Dr. Nour Mahmoud

Implemented by

20205022	Maram Alaa Eldein Abd Elgayed
20205025	Souhila Sherif Mahmoud
20205026	Tasneem Yousry Abd Almageed

Graduation Project
Academic Year 2023-2024
Final Documentation

Table of Contents

Chapter 1: Introduction.....	5
1.1 Motivation	5
1.2 Background	6
1.3 Problem definition.....	6
1.4 Our solution	8
1.5 Gantt chart.....	9
1.6 Project development methodology.....	10
1.7 Used tools in the project	13
1.8 Report organization	14
Chapter 2: Related Work.....	14
Chapter 3: System Analysis	17
3.1 Project specifications	17
3.1.1 System Architecture.....	17
3.1.2 Stakeholders.....	18
3.1.3 Functional requirements.....	19
3.1.4 Non-functional requirements.....	21
3.2 Use case diagrams.....	22
Chapter 4: System Design	23
4.1 System component diagram	23
4.2 System class diagrams.....	24
4.3 Sequence diagrams.....	25
4.3.1. Sign up	25
4.3.2. Login	25
4.3.3. Cross-platform Profile search.....	26
4.3.4. Get user tweets.....	26
4.3.5. Get keyword tweets	27

4.3.6. Get User Facebook Data.....	27
4.3.7. Search Keyword in Facebook.....	28
4.3.8. Search LinkedIn (By Username)	28
4.3.9. View History.....	29
4.4 Project ERD.....	29
4.5 System GUI design.....	30
Chapter 5: Implementation and testing.....	36
5.1 Implementation.....	36
5.1.1 Frontend	36
5.1.2 Backend	37
5.1.3 Database.....	49
5.2 Testing	50
References	56

List of Tables

Table 1 Related work	15
Table 2 Testing	51

List of Figures

Figure 1 Gantt chart	9
Figure 2 System Architecture	17
Figure 3 Use case diagram	22
Figure 4 System component diagram.....	23
Figure 5 Class diagram	24
Figure 6 Sign up sequence diagram.....	25
Figure 7 Log in sequence diagram.....	25
Figure 8 cross platform sequence diagram	26

Figure 9 Get user tweets sequence diagram.....	26
Figure 10 Get keyword tweets sequence diagram.....	27
Figure 11 Get Facebook user data sequence diagram.....	27
Figure 12 search keyword in Facebook sequence diagram.....	28
Figure 13 Search LinkedIn by username sequence diagram.....	28
Figure 14 View history sequence diagram.....	29
Figure 15 ERD	29
Figure 16 Sign up GUI	30
Figure 17 Log in GUI.....	30
Figure 18 Services page GUI	31
Figure 19 Cross platform GUI	31
Figure 20 About us page GUI.....	32
Figure 21 Statistics page GUI	32
Figure 22 History page GUI.....	33
Figure 23 Profile page GUI	33
Figure 24 LinkedIn search GUI	34
Figure 25 Facebook search GUI	34
Figure 26 Twitter search GUI	35
Figure 27 Frontend file hierarchy	36
Figure 28 Node.js Backend ex.1	37
Figure 29 Node.js Backend ex.2	38
Figure 30 Node.js Backend ex.3	38
Figure 31 Python Backend ex.1	39
Figure 32 Python Backend ex.2	40
Figure 33 Python Backend ex.3	41
Figure 34 Python Backend ex.4	42
Figure 35 Python Backend ex.5	42
Figure 36 Python Backend ex.6	43
Figure 37 Python Backend ex.7	43
Figure 38 Python Backend ex.8	44
Figure 39 Python Backend ex.9	45
Figure 40 Python Backend ex.10	45
Figure 41 Python Backend ex.11	46
Figure 42 Python Backend ex.12	46

Figure 43 Python Backend ex.13	47
Figure 44 Python Backend ex.14	47
Figure 45 Python Backend ex.15	48
Figure 46 Python Backend ex.16	48
Figure 47 Cloud database collections	49
Figure 48 Test case 1	52
Figure 49 Test case 2.1	52
Figure 50 Test case 2.2	53
Figure 51 Test case 3.1	53
Figure 52 Test case 3.2	54
Figure 53 Test case 3.3	54
Figure 54 Test case 3.4	55
Figure 55 Test case 4.1	55
Figure 56 Test case 4.2	55

Chapter 1: Introduction

1.1 Motivation

You can tell a lot about a person from their social media accounts. That would've been debatable if there weren't 2 billion daily active users on Facebook, or over 1 billion daily users on LinkedIn and 335 million users on Twitter. But the question remains, "How much is "a lot" in this context?", "Should I be concerned?", "Don't those platforms have sophisticated, multi-million-worth security measures?". The short answer is yes. The long answer we will explore in this project. Our main goal is to try and scrape data from the above-mentioned platforms using basic tools, minimum technical skills, and little to no paid services. We do that to prove that anyone can do it and hopefully raise awareness of the dangers of having the habit of revealing too much about yourself online. Not to mention that our project can be used for other purposes such as OSINT, threat intelligence, and data analysis.

For this project, we will use Python and Nodejs for data extraction and automation and then save that data in a MongoDB database for faster access in subsequent requests. Our website has numerous filters/search methods the user can choose from to get the most accurate results.

1.2 Background

While conducting searches for ideas related to the cyber security field, we stumbled upon the interesting concept of OSINT (Open-Source Intelligence) , with a simple search on LinkedIn more than 800 OSINT related jobs was found in the US with job titles ranging from intelligence consultant , OSINT investigator and even Interpol related positions, but with the existence of billions of social media users around the world who would also occasionally want to search for specific data with little to no technical background , we decided to provide a tool that would be of help to both technical and non-technical users.

It is A website designed with the aim of making open-source intelligence an easier and more accessible process, it focuses on social media public data collection. Users can search using a person's name to find all possible results of accounts holding the same name, and then can utilize this information to start collecting data from respective user's accounts, they can choose to collect all available data or start using filters to extract what they believe is valuable for their search.

The focus of this project is web scraping through different social media platforms using different languages and libraries, then representing the findings in an easy and user-friendly interface.

1.3 Problem definition

In today's digital landscape, social media platforms have become integral to our daily lives, with millions of users engaging in various activities that generate a vast amount of valuable data. This data, created unconsciously by users through their interactions, posts, and shared content, holds immense potential for insights across numerous fields such as marketing, sociology, psychology, and more. As the number of social

media users continues to rise, with projections indicating significant growth in the coming years, the need to harness this data becomes increasingly vital.

Upon analysing current statistics and reports, we recognized a critical opportunity to develop a solution that effectively gathers and utilises this wealth of information.

However, our preliminary research into existing tools and services revealed several limitations. While there are numerous websites, libraries, and tools that partially address the need for social media data collection, they often fall short in key areas:

- 2 **Platform-Specific Limitations:** Many existing solutions focus on a single social media platform, thereby missing out on the broader picture that can be obtained by aggregating data from multiple sources.
- 3 **Lack of Aggregation:** Even tools that do collect data from multiple platforms often fail to aggregate this data into a cohesive and unified format, limiting the ability to perform comprehensive analyses.
- 4 **User Accessibility and Usability:** The existing tools are often designed for developers or data scientists, lacking user-friendly interfaces that allow a broader audience to access and utilise the data effectively.
- 5 **Data Filtering and Customization:** There is a noticeable gap in providing robust filtering and customization options, which are essential for users who need specific types of data for targeted analyses.

Driven by these insights and the evident gap in the market, we decided to embark on developing an all-in-one web application that addresses these shortcomings. Our project, Xtract, aims to provide a comprehensive solution for collecting, aggregating, and analysing public data from various social media platforms. Key features of our application include:

- **Multi-Platform Integration:** Seamless integration with major social media platforms to gather diverse datasets.
- **Data Aggregation:** Combining data from different platforms into a single, cohesive repository for more comprehensive analysis.

- **User-Friendly Interface:** Designing intuitive and accessible interfaces to ensure usability by a wide range of users, from researchers to marketers.
- **Advanced Filtering and Customization:** Offering powerful tools to filter, customize, and extract the precise data needed for specific purposes.

By addressing these critical gaps, Xtract aims to revolutionise the way social media data is collected and utilised, unlocking new possibilities for insights and applications. This project represents a significant step forward in making social media data more accessible, usable, and valuable for various stakeholders.

1.4 Our solution

The primary objective of our project, Xtract, is to develop a robust and comprehensive web application that seamlessly integrates with multiple social media platforms to collect, aggregate, and analyse publicly available data. By addressing the limitations of existing tools, Xtract aims to revolutionise the way users access and utilize social media data, transforming it into actionable insights. Our solution will be designed with a focus on user-friendliness, ensuring that a wide range of users, from researchers and data scientists to marketers and casual users, can easily navigate and harness the power of the application.

Xtract will feature multi-platform integration, enabling it to gather data from major social media sites like Facebook, Twitter, Instagram, LinkedIn, and more. This data will be aggregated into a single, cohesive repository, allowing for a more holistic view and comprehensive analysis. Our application will provide advanced filtering and customization options, empowering users to extract the precise information they need based on various criteria such as time frame, location, keywords, and user demographics.

In addition to these core functionalities, Xtract will incorporate sophisticated data analysis and visualization tools, presenting the collected data in intuitive formats such as graphs, charts, and interactive dashboards. This will facilitate deeper insights and enable users to identify trends, patterns, and correlations with ease. Our application

will also prioritize data privacy and ethical considerations, ensuring that all collected data is handled in compliance with relevant regulations and best practices.

By achieving these objectives, Xtract aims to fill the current market gap and provide a comprehensive solution that not only simplifies the process of social media data collection but also enhances its utility and accessibility. Ultimately, our goal is to empower users with the tools they need to unlock the full potential of social media data, driving informed decision-making and fostering innovative applications across various domains.

1.5 Gantt chart

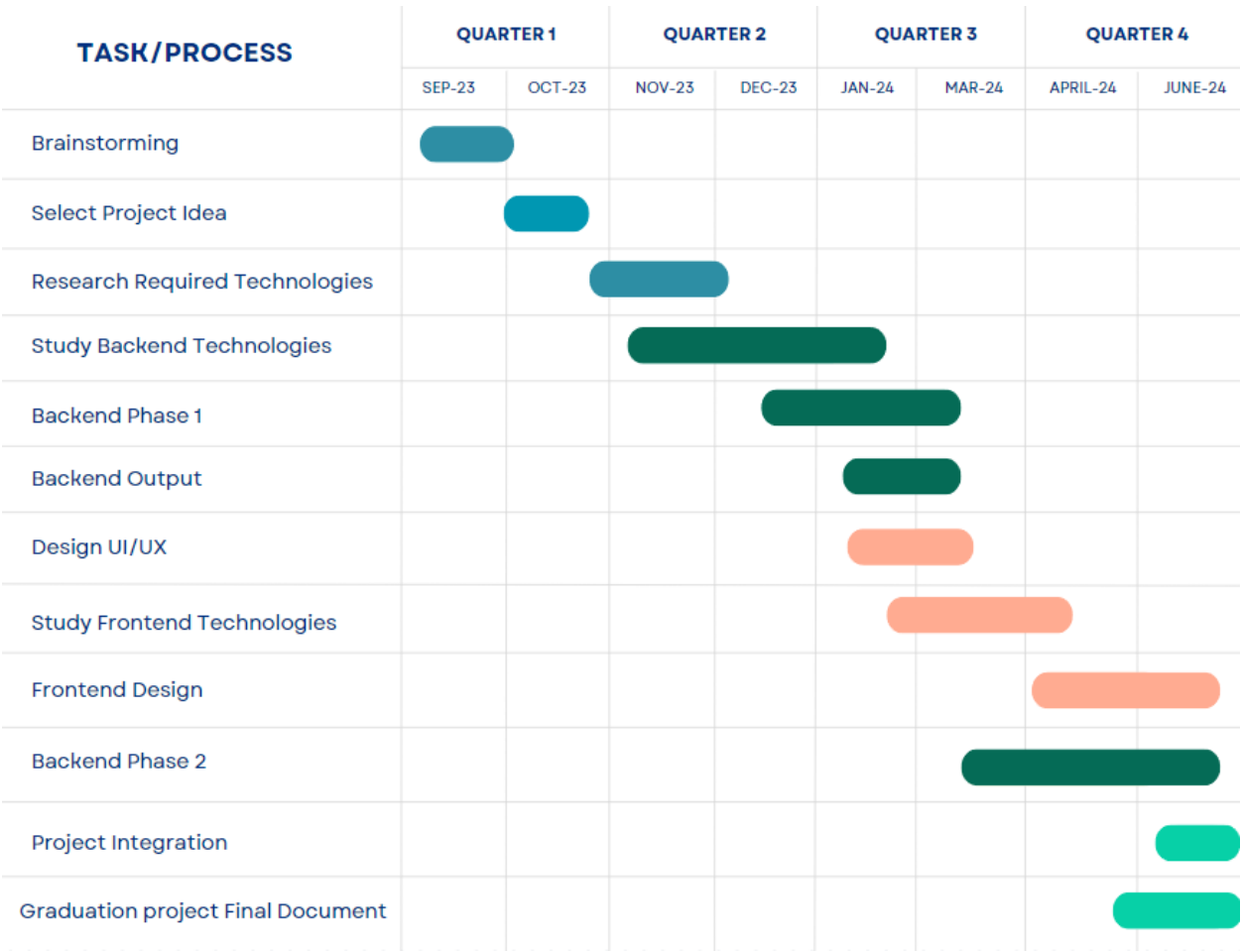


Figure 1 Gantt chart

1.6 Project development methodology

Agile framework

Our development process is grounded in the principles of the Agile framework, which allows us to be flexible, iterative, and responsive to change. Agile helps us maintain a steady pace of development while adapting to new requirements and feedback.

- **Sprints:** We operate in bi-weekly sprints. Each sprint begins with a planning session to define clear objectives and ends with a review meeting to evaluate progress and gather feedback.
- **Sprint Planning:** During sprint planning, we prioritize the backlog, allocate tasks based on team capacity, and set sprint goals. This ensures everyone is on the same page and working towards common objectives.
- **Sprint Reviews:** At the end of each sprint, we hold a review session to showcase completed work, discuss what was achieved, and gather feedback from stakeholders. This helps us ensure we're meeting user needs and project goals.
- **Reviews and Retrospectives:** At the end of each cycle, we review completed work, gather feedback, and reflect on our processes to identify areas for improvement.

Collaboration and Communication

Effective collaboration and communication are critical to our success. We use a variety of tools and practices to ensure our team stays connected, informed, and efficient.

- **Version Control with Git:** Git allows us to manage our codebase collaboratively. Branching, merging, and pull requests help us handle changes smoothly and maintain code quality.

- **Code Reviews:** Regular code reviews are integral to our process. They help us maintain high code quality, share knowledge, and identify potential improvements. Peer feedback ensures robust and maintainable code.

Technology Stack

Our diverse technology stack enables us to build a powerful and flexible platform. Each component is chosen to meet specific needs and optimize performance.

- **Node.js Backend:** Node.js is used for its non-blocking, event-driven architecture, which makes it ideal for handling asynchronous operations and scalable network applications.
- **Python Backend:** Python is used for data processing, automation, and certain backend tasks. Its simplicity and rich ecosystem of libraries make it an excellent choice for various functionalities.
- **Vue.js Front-End:** Vue.js is our chosen framework for building interactive and responsive user interfaces. Its component-based architecture allows for modular and maintainable code.
- **Automation and Scraping Tools:** Puppeteer and Selenium are employed for web scraping and automation. These tools help us collect and process data from various social media platforms efficiently.

Team Responsibilities

To ensure focused expertise and efficient development, each team member is responsible for a specific social media platform. This specialization allows us to leverage in-depth knowledge and tailor our approaches to the unique aspects of each platform.

- **Platform Assignments:** Team members are assigned to different social media platforms, ensuring comprehensive coverage and specialized knowledge. This structure enables us to tackle platform-specific challenges effectively and optimize data collection.

User-Centred Design

Our development process prioritizes the needs and experiences of our users. We employ a user-centred design approach to ensure our platform is intuitive, accessible, and meets user expectations.

- **User Research:** We conduct thorough user research to understand the behaviours, needs, and pain points of our target audience. This informs our design decisions and helps us create user-friendly features.
- **Iterative Development:** We continuously iterate on our features based on testing results. This ensures our platform evolves in line with user needs and provides the best possible experience.

Quality Assurance

Quality is paramount in our development process. We have robust QA practices in place to ensure our platform is reliable, performant, and bug-free.

- **Manual Testing:** In addition to automated tests, our QA team conducts manual testing to cover edge cases and focus on user experience. This comprehensive approach ensures high quality.
- **Performance Monitoring:** We use monitoring tools to track the performance and stability of our platform. This allows us to identify and address issues proactively, ensuring a smooth user experience.

Continuous Improvement

Continuous improvement is a core principle of our methodology. We regularly review our processes, tools, and techniques to identify areas for enhancement and stay updated with industry best practices.

- **Process Reviews:** Regular process reviews help us identify inefficiencies and implement improvements. This ensures our workflows are optimized and effective.

- **Feedback Loop:** We maintain a feedback loop with users and stakeholders to ensure our platform meets their needs and expectations. This ongoing dialogue helps us refine our product and deliver value.

1.7 Used tools in the project

A variety of programming languages, frameworks, and applications are used to build this website which are listed as follows:

Figma: which is a collaborative web application for interface design that we used in the UI/UX design process.

HTML&CSS: HTML is the standard markup language for documents designed to be displayed in a web browser. It defines the content and structure of web content accompanied by CSS that is used to specify the presentation and styling of a document written in a markup language which is HTML in our case.

Vue.js: is open-source frontend JavaScript framework for building user interfaces and single-page applications, the version used in this website implementation is Vue 3.

JavaScript: is a programming language and core technology of the World Wide Web, alongside HTML and CSS, it is used in our frontend in the context of using Vue.js and also used in parts of the backend by using the framework Node.js.

Python: Python is a high-level, general-purpose programming language that is used alongside JavaScript on our website to build the backend.

Node.js: is an open-source JavaScript runtime environment, it lets developers use JavaScript to write command line tools and server-side scripting, which means that it is used as the framework to write the JavaScript parts of our backend.

Express.js: it is the framework used on top of Node.js to write the necessary API calls that allows the frontend to communicate with our Node.js backend.

Axios: is the JavaScript library we use in our vue.js frontend to make http requests to communicate with the backend.

Puppeteer: is a Node.js library which provides a chrome browser to run and test your JavaScript codes and requests.

Selenium: is an open-source, automated testing tool used to test web applications across various browsers. It can be used with various languages but in the context of this website it is used with Python.

MongoDB: is a cross-platform, document-oriented database program. Classified as a NoSQL database product, it is used on this website to store the resulted JSON data.

1.8 Report organization

For the rest of the document we will be delving into more details about Xtract as a project , In Chapter 2 we discuss similar solutions that exist in the market and how Xtract is different, in Chapter 3 we get into the actual system analysis that draws the big picture of how the project looks like , while in Chapter 4 we describe the system design that will then be shown as actual implementation in Chapter 5 that also includes project testing steps and results.

Chapter 2: Related Work

The comparison table provides a comprehensive overview of five web scraping platforms: Xtract, Octoparse, ParseHub, Social Scraper, and Twint. Each row represents a key feature or aspect of these platforms, ranging from their ability to target social media platforms to their pricing structure. By evaluating these features side by side, we can easily identify the strengths and weaknesses of each platform, ultimately aiding in their decision-making process when choosing a web scraping solution

Feature	Xtract	Octoparse	ParseHub	Social Scraper	Twint
Support Different Social Media Scraping	✓	✓	✓	×	×
Targets Social Media Platforms	✓	✓	✓	✓	✓
User-Friendly GUI	✓	✓	✓	×	×
Data Exports Formats (JSON, CSV)	✓	✓	✓	✓	✓
Transparency in Data Handling	✓	×	✓	×	✓
Price	✓	×	×	✓	✓

Table 1 Related work

Feature Descriptions

- **Support Different Social Media Scraping:** Indicates the platform's ability to scrape data from various social media platforms, essential for comprehensive data gathering.
- **Targets Social Media Platforms:** Shows whether the platform is specifically designed to scrape social media data, which is crucial for targeted data collection.
- **User-Friendly GUI:** A user-friendly graphical user interface makes it easier for users of all skill levels to navigate and use the platform effectively.
- **Data Exports Formats (JSON, CSV):** The ability to export data in common formats like JSON and CSV allows for easier data manipulation and analysis.
- **Transparency in Data Handling:** Transparency in how data is handled and processed ensures trust and compliance with data protection regulations.
- **Price:** Indicates the cost-effectiveness of the platform, an important factor for budget-conscious users.
- **Statistical Analysis of CSV Data (Planned):** Xtract aims to provide statistical analysis features for data exported in CSV format. This capability will allow users to gain deeper insights and extract meaningful patterns from their scraped data, enhancing the platform's analytical capabilities.

These planned features position Xtract as a more comprehensive and forward-looking solution for users seeking a web scraping platform that not only meets their current needs but also offers advanced analytical capabilities and future enhancements.

Chapter 3: System Analysis

3.1 Project specifications

3.1.1 System Architecture

The system is a client-server layered architecture.

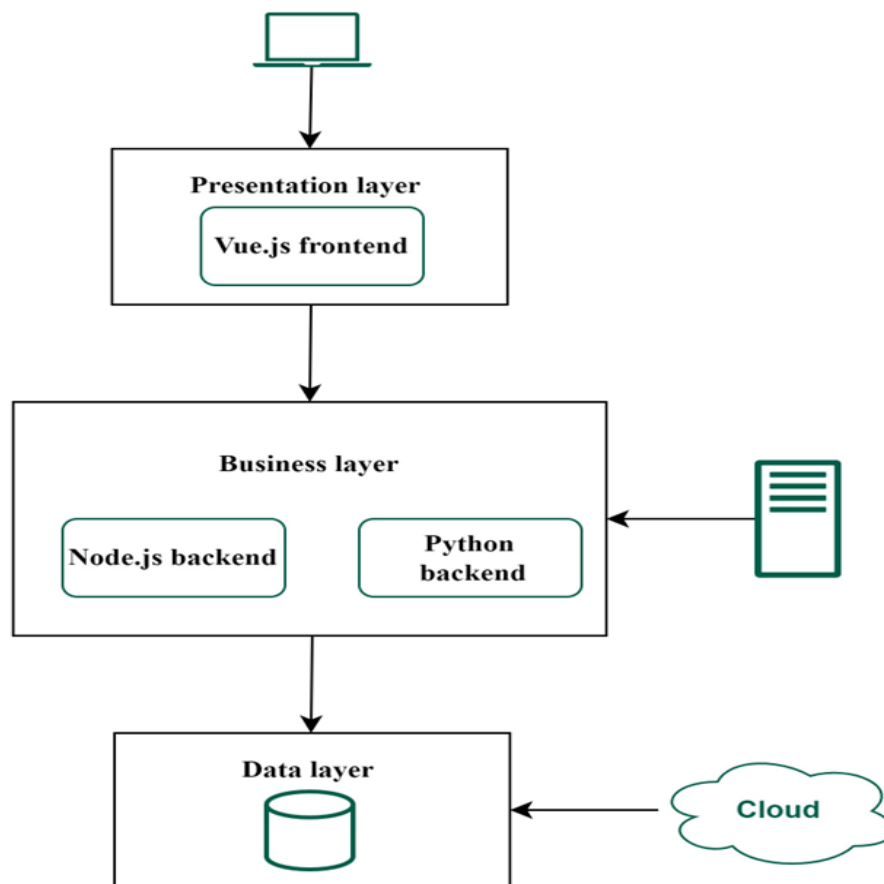


Figure 2 System Architecture

3.1.2 Stakeholders

- **Users:**

- **Researchers and Data Scientists:** Seek comprehensive and reliable data for academic studies, market research, or trend analysis. They value robust data collection methods, advanced analytics capabilities, and customizable filtering options.
- **Marketers:** Use social media insights to refine advertising strategies, understand audience behaviour, and measure campaign effectiveness. They prioritize user-friendly interfaces, real-time data updates, and integration with marketing tools.
- **Casual Users:** Interested in exploring social media trends, discovering content, or monitoring personal or professional profiles. They appreciate intuitive navigation, simplified data visualization, and privacy controls.

- **Businesses and Organizations:**

- **Market Research and Strategy Teams:** Depend on accurate and timely social media data to identify market trends, consumer preferences, and competitive insights. They value comprehensive data coverage across multiple platforms and actionable insights.
- **Digital Marketing Agencies:** Utilize social media analytics to optimize client campaigns, enhance audience engagement, and measure return on investment (ROI). They seek integration capabilities with existing marketing platforms and customizable reporting features.

- **Developers:**

- With a user-friendly UI and comprehensive API documentation, Xtract caters to developers with varying levels of experience in data extraction and analysis. The platform simplifies the integration of social media data into applications, enabling developers to build data-rich solutions without needing extensive expertise in web scraping or data aggregation. Xtract's robust API and flexible configuration options allow developers to focus on creating innovative applications and features with confidence.

- **Investors and Stakeholders:**

- Investors and stakeholders are interested in the growth potential and market positioning of Xtract. They evaluate the platform's innovative features, market demand, and competitive advantage. Xtract's focus on user-friendliness, advanced data analysis, and compliance with data privacy regulations positions it as a valuable investment opportunity. Comprehensive reports and insights provided by Xtract give investors the confidence to support and fund the platform's development and expansion.

3.1.3 Functional requirements

1. Basic Requirements:

1.1 Login and Registration:

- The user should be able to register with a username and password.
- The user can login if he already has an account registered with us.
- The user doesn't have to login every time he visits the website, the user's session is stored and saved with an expiry timer.

1.2 User Main operations:

- The user can choose from the available services (LinkedIn, Twitter, Facebook, cross-platform search).

Twitter Search Service:

- The user can search in twitter by a keyword and get the tweets where that keyword is mentioned or by a username and retrieve the tweets of that user.
- The user can filter the results by date, time, keyword and number of tweets.

LinkedIn Search Service:

- The user can search in LinkedIn by a username and retrieve profile attributes such as account name, subtitle, about, education and experience section and posts.
- The user can control what information he gets and filter the posts by date, keyword and amount.
- The user can search by keyword and get posts where that keyword appeared.
- The user can filter the results by date and amount.

Facebook Search Service:

- The user can search in Facebook with a username and get information about the account such as education, work, living location, contact info, relationship, bio, events, and other basic info.
- The user can also get the account's photos and posts.
- The user can search by a keyword and get public posts where that keyword is mentioned.
- The user can filter posts by date and amount.

Cross-Platform Search Service:

- The user can look for a person in the three platforms at once by entering the person's name.

1.3 View and edit profile:

- The user can change his password.
- The user can view his search history.

2. I/O Requirement:

- The user can download the results as CSV or JSON files.

3. Statistical and Graphical Requirements:

- The user can view different statistics about his account and our website

3.1.4 Non-functional requirements

- **User friendly:** it is a website with easy-to-understand options and filters.

- **Scalability:** the layered nature of the architecture makes it easy to add different components and services in the future.

- **Performance:** the data gathering process takes relatively more time but should not exceed a couple of minutes, while the retrieval of data from MongoDB should take from 3 to 5 seconds, also MongoDB allows large volumes of data.

- **Security:** the users should be able to authenticate as legitimate users, and the website code should be immune to known vulnerabilities like XSS and CSRF.

3.2 Use case diagrams

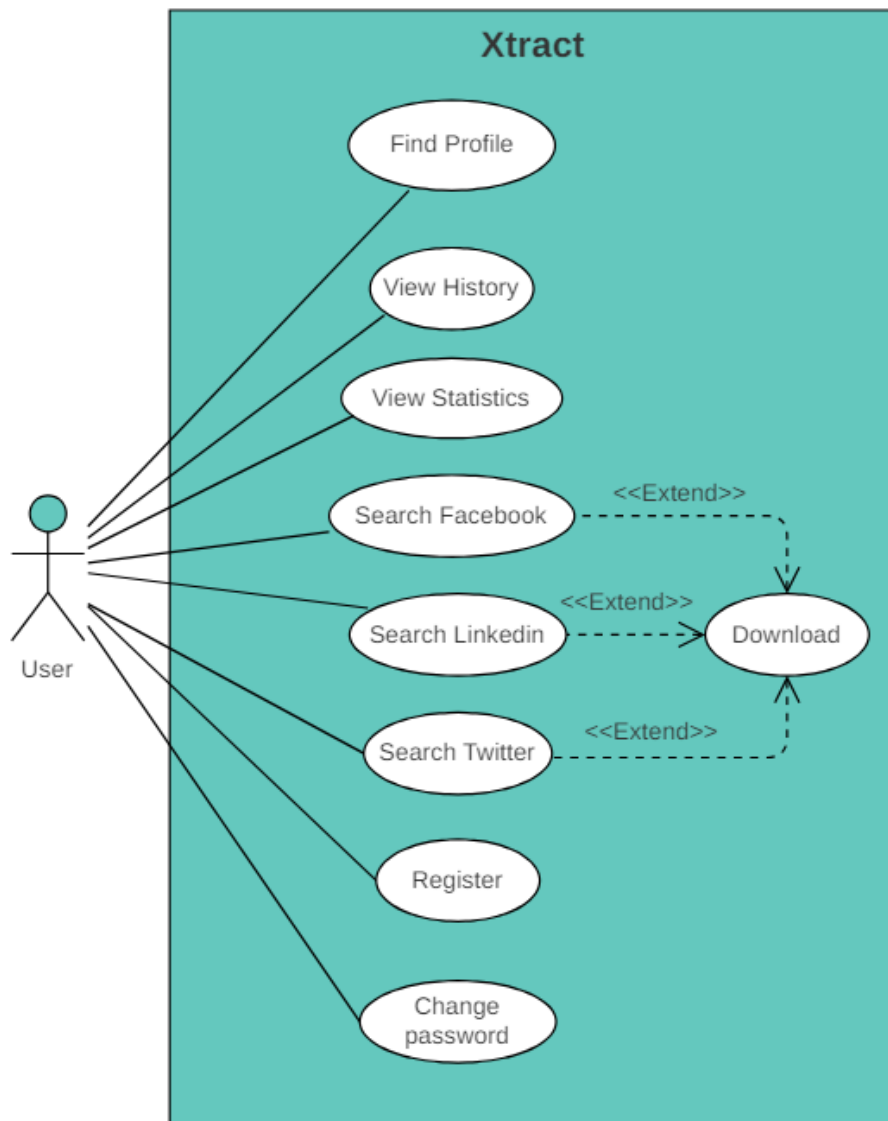


Figure 3 Use case diagram

Chapter 4: System Design

4.1 System component diagram

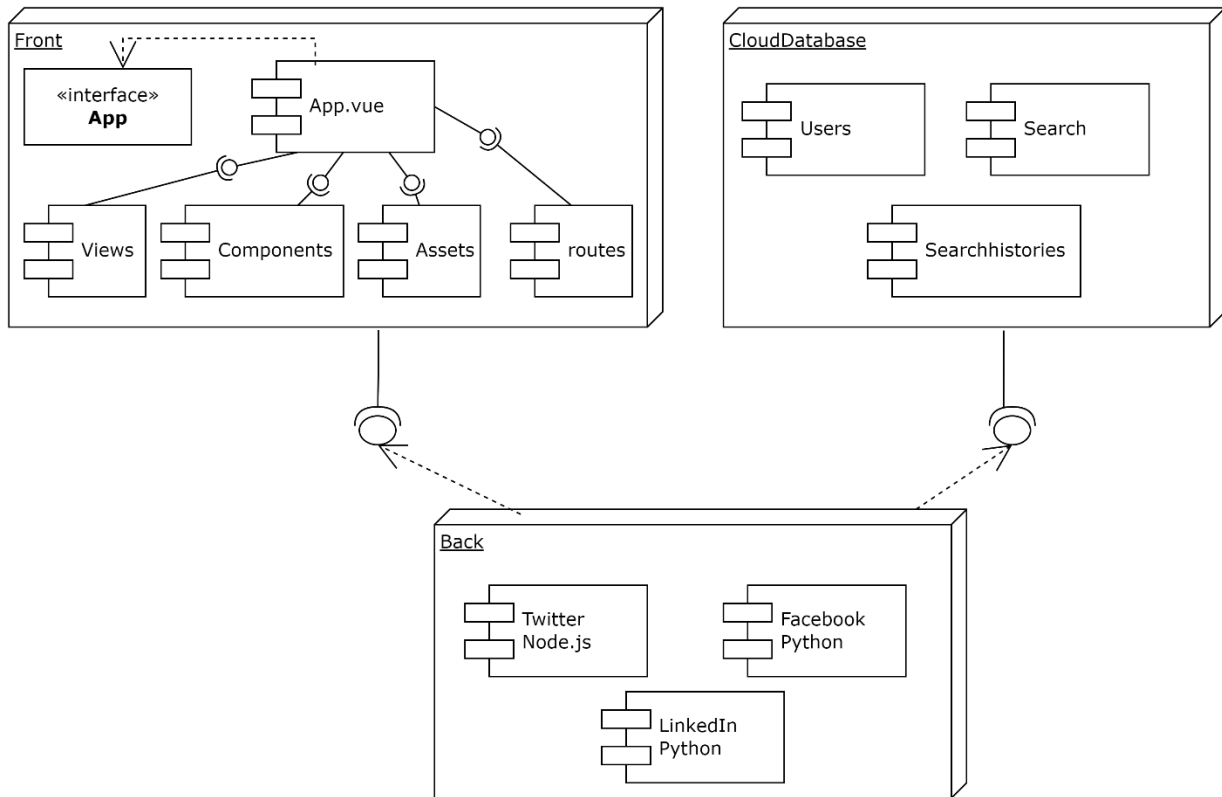


Figure 4 System component diagram

4.2 System class diagrams

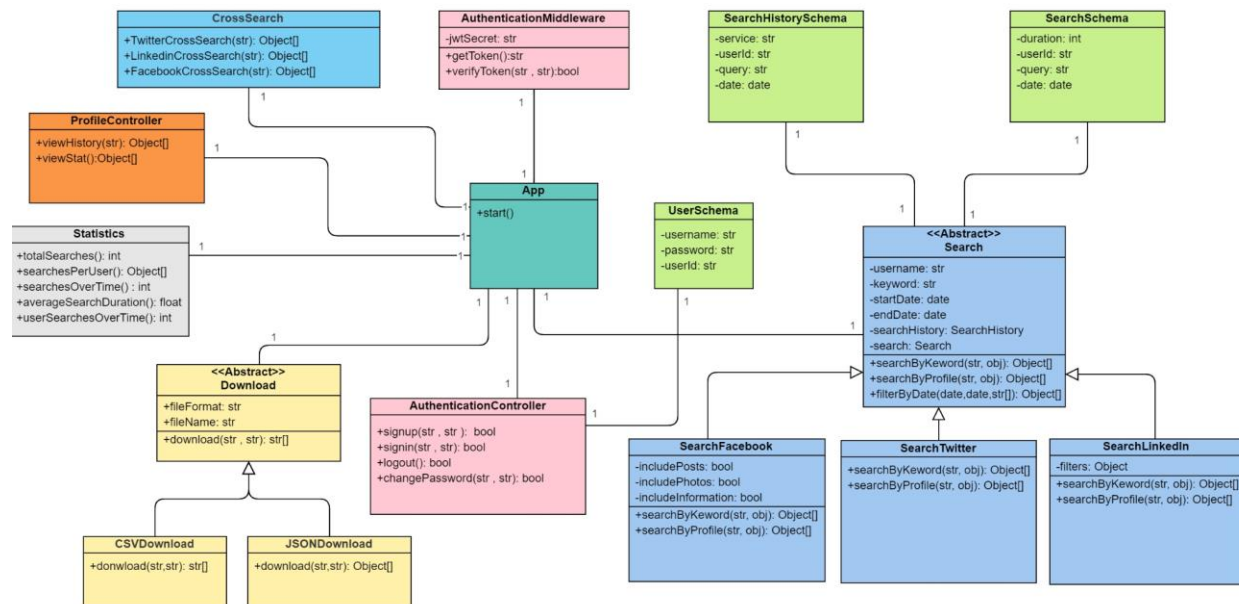


Figure 5 Class diagram

4.3 Sequence diagrams

4.3.1. Sign up

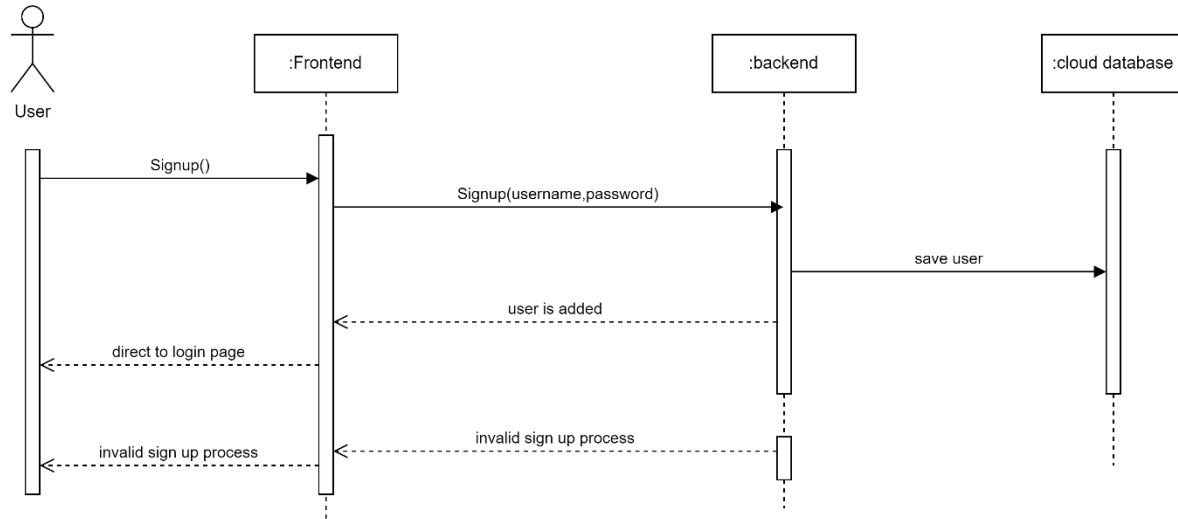


Figure 6 Sign up sequence diagram

4.3.2. Login

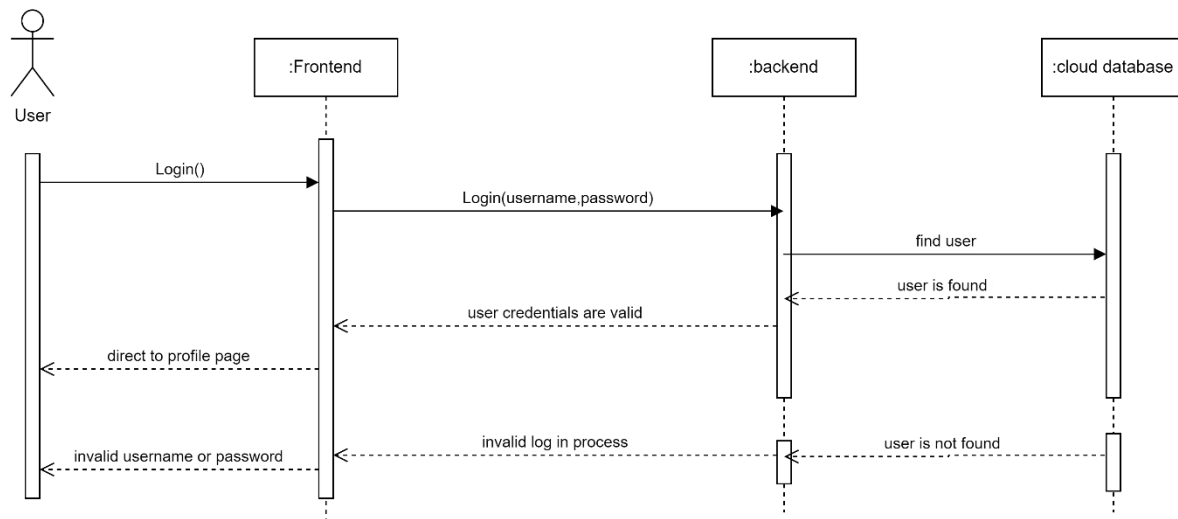


Figure 7 Log in sequence diagram

4.3.3. Cross-platform Profile search

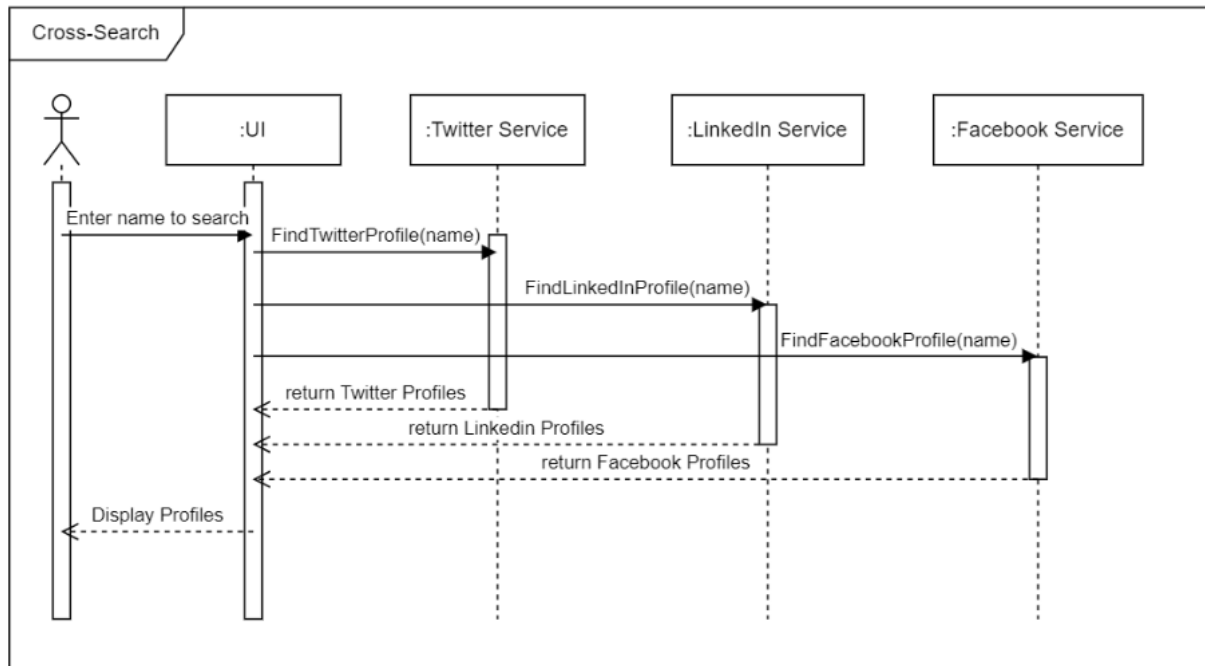


Figure 8 cross platform sequence diagram

4.3.4. Get user tweets

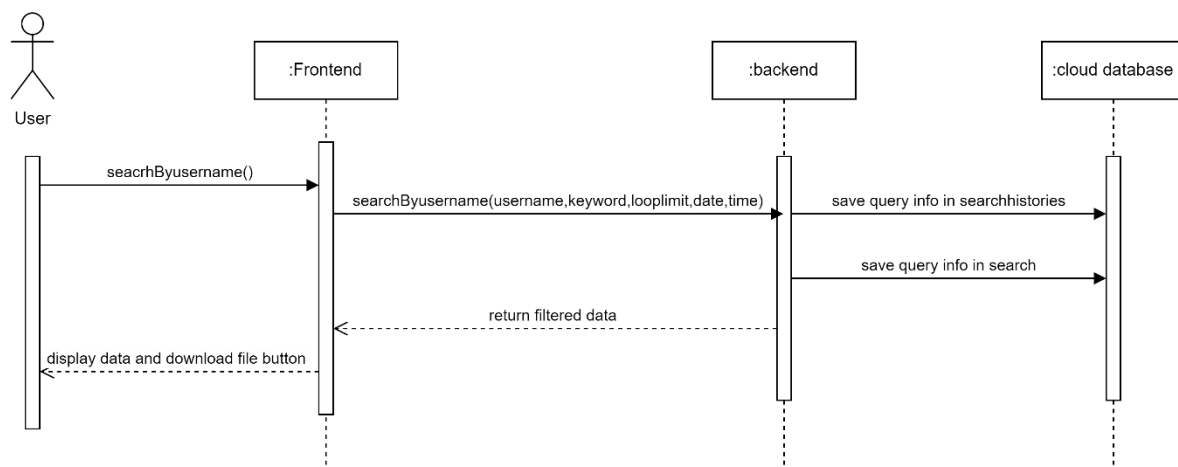


Figure 9 Get user tweets sequence diagram

4.3.5. Get keyword tweets

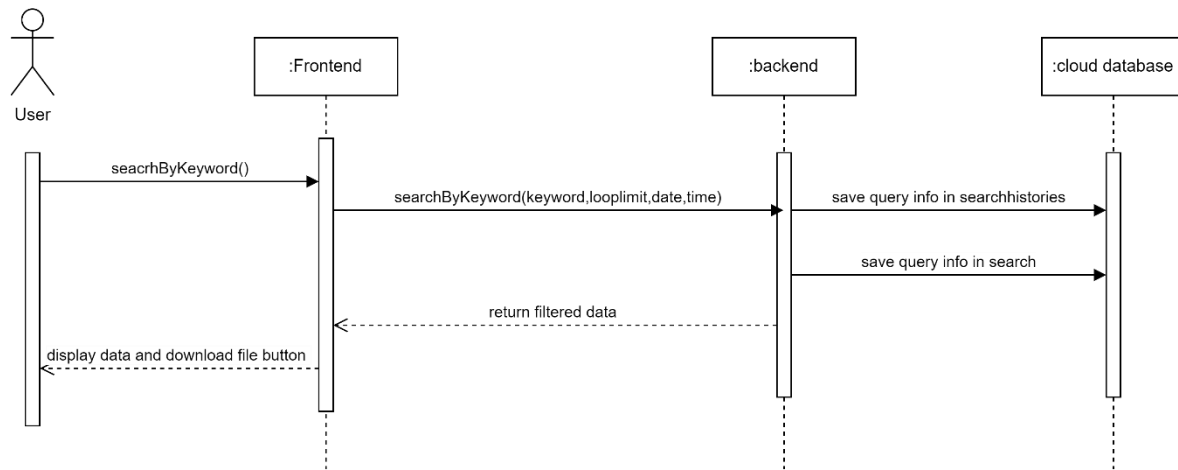


Figure 10 Get keyword tweets sequence diagram

4.3.6. Get User Facebook Data

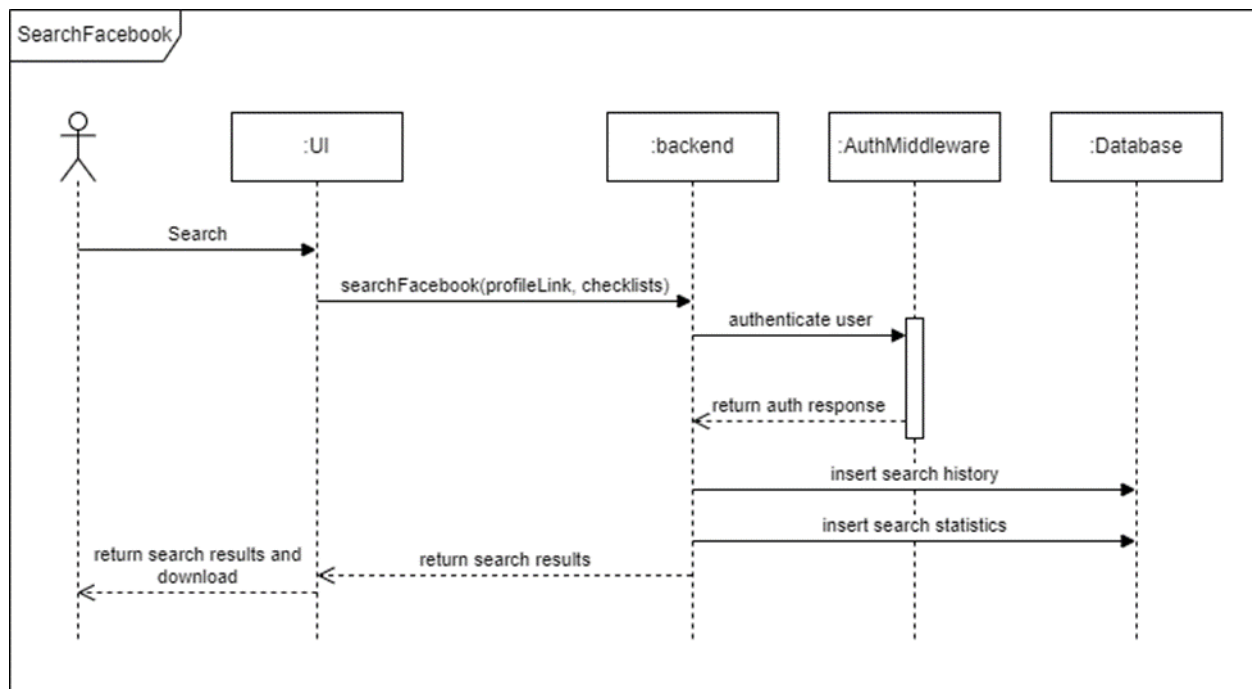


Figure 11 Get Facebook user data sequence diagram

4.3.7. Search Keyword in Facebook

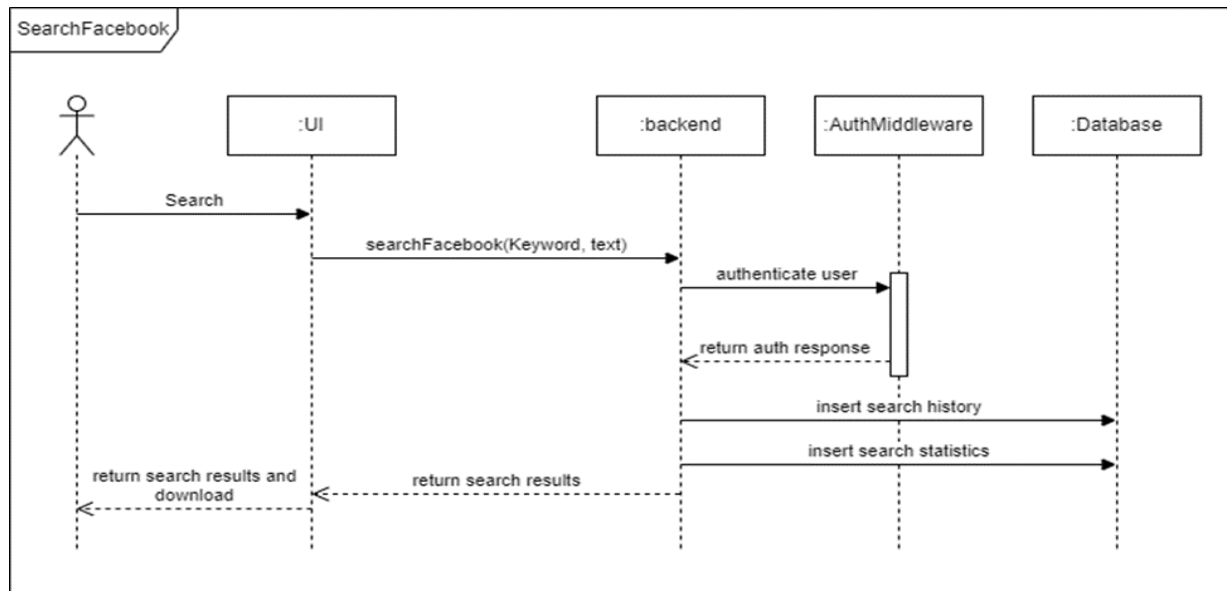


Figure 12 search keyword in Facebook sequence diagram

4.3.8. Search LinkedIn (By Username)

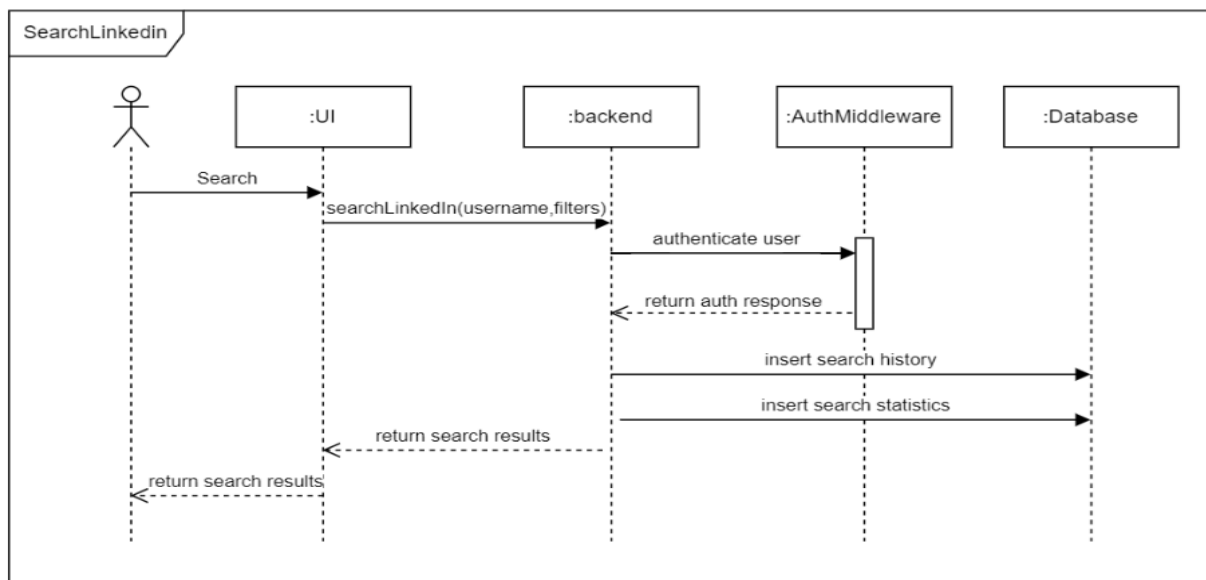


Figure 13 Search LinkedIn by username sequence diagram

4.3.9. View History

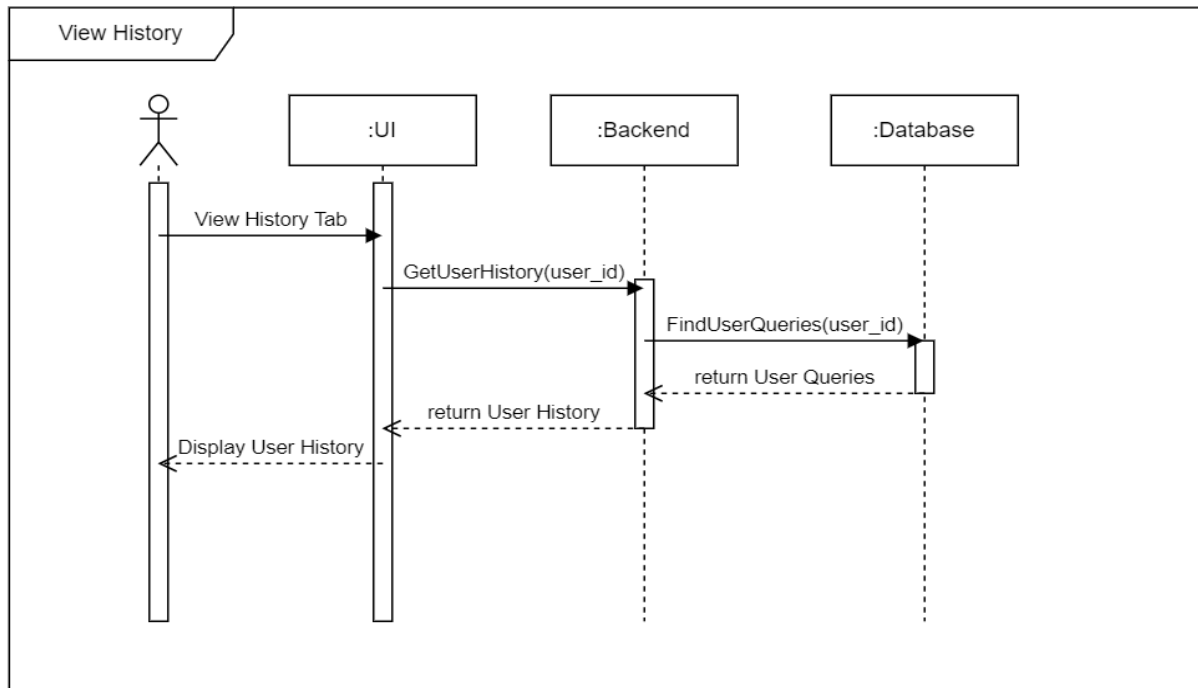


Figure 14 View history sequence diagram

4.4 Project ERD

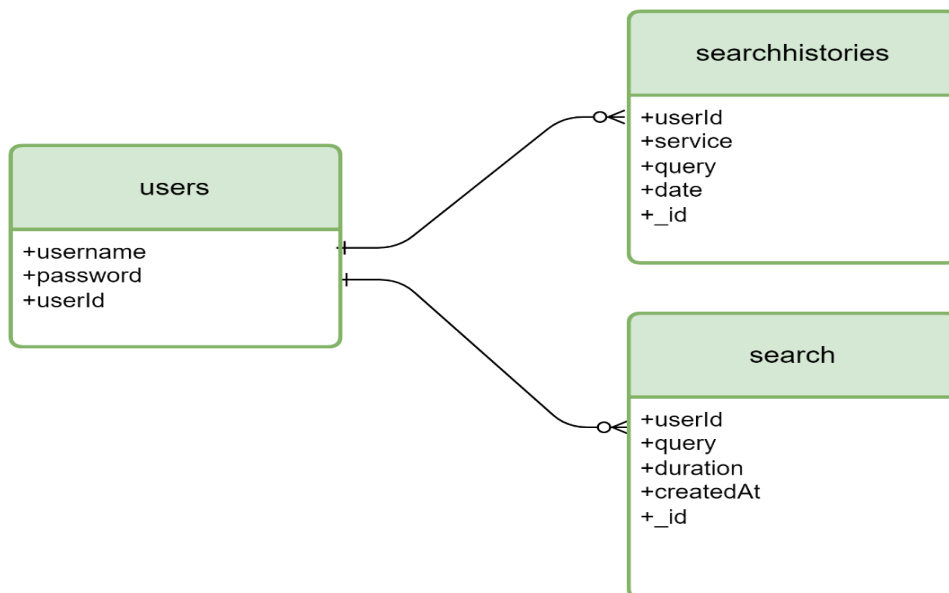
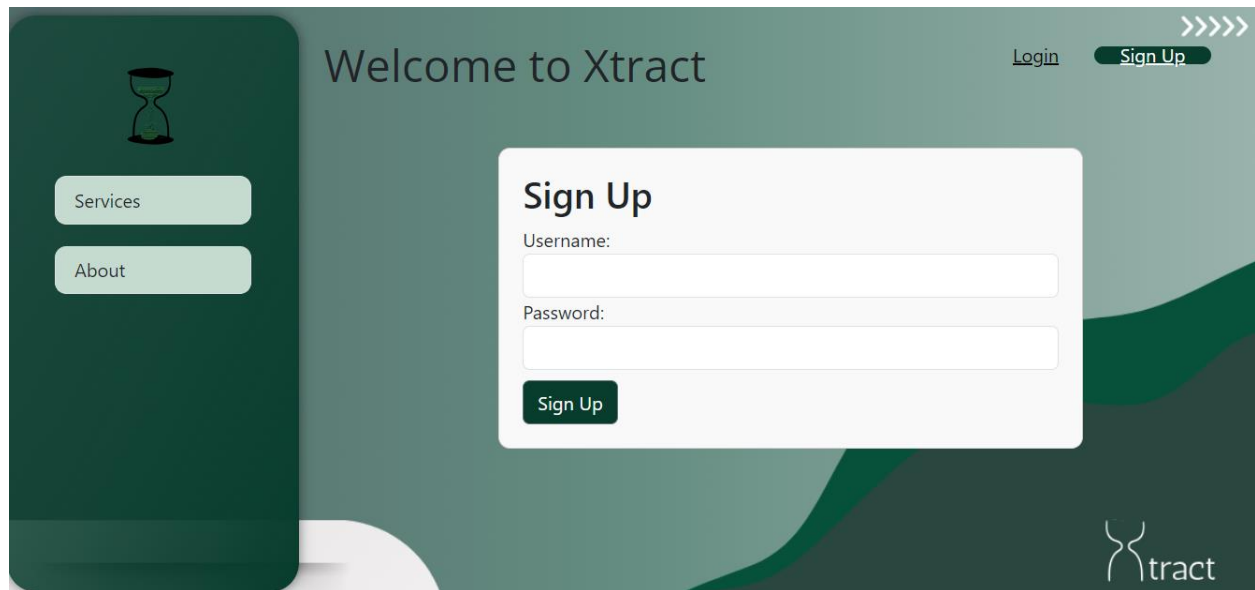


Figure 15 ERD

4.5 System GUI design



The image shows a web application interface for signing up. On the left is a dark green sidebar with an hourglass icon and two buttons: "Services" and "About". The main area has a light green background with the text "Welcome to Xtract" at the top. In the top right corner, there are links for "Login" and "Sign Up", with "Sign Up" being highlighted. A white "Sign Up" form is centered, containing fields for "Username:" and "Password:", and a "Sign Up" button. The "Xtract" logo is in the bottom right corner.

Services

About

Welcome to Xtract

Login Sign Up

Sign Up

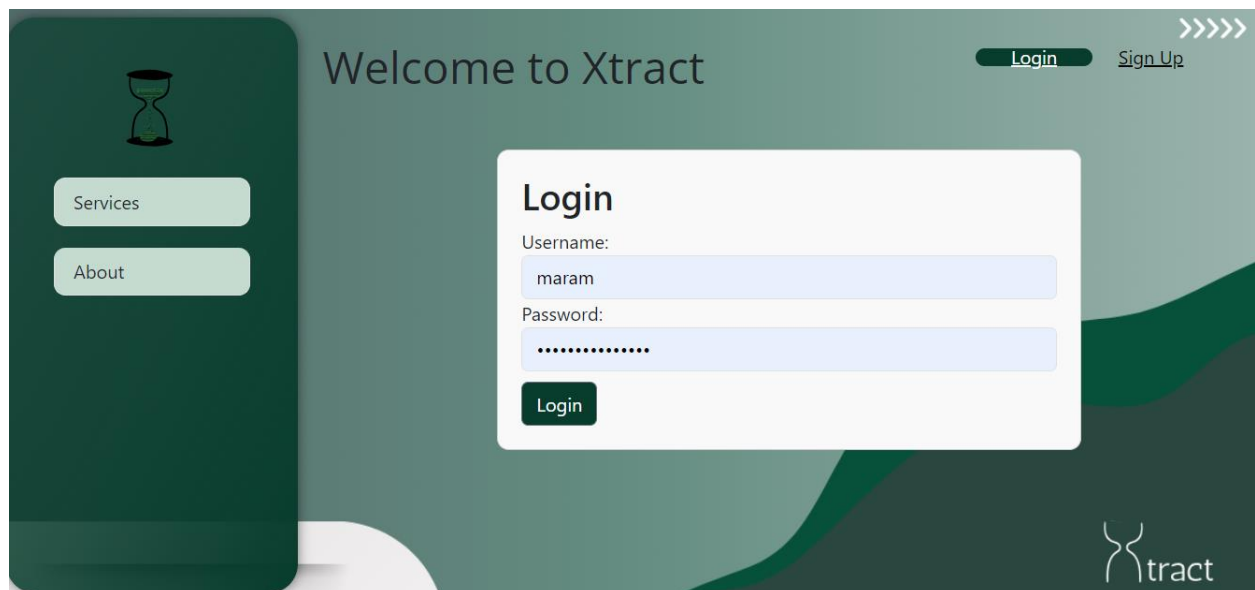
Username:

Password:

Sign Up

Xtract

Figure 16 Sign up GUI



The image shows a web application interface for logging in. It has the same layout as the sign-up page, but the "Sign Up" button in the top right is highlighted instead of the "Login" button. The white form in the center is titled "Login" and contains fields for "Username:" (with the text "maram" entered) and "Password:" (with masked characters "*****" entered), and a "Login" button. The "Xtract" logo is in the bottom right corner.

Services

About

Welcome to Xtract

Login Sign Up

Login

Username:

maram

Password:

Login

Xtract

Figure 17 Log in GUI

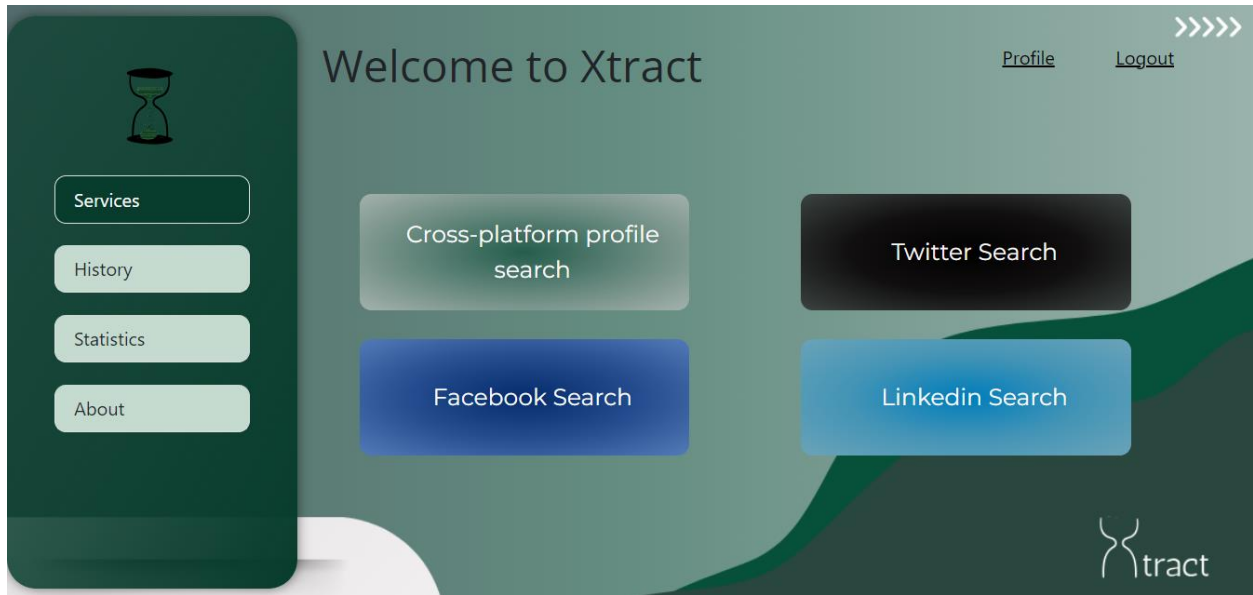


Figure 18 Services page GUI

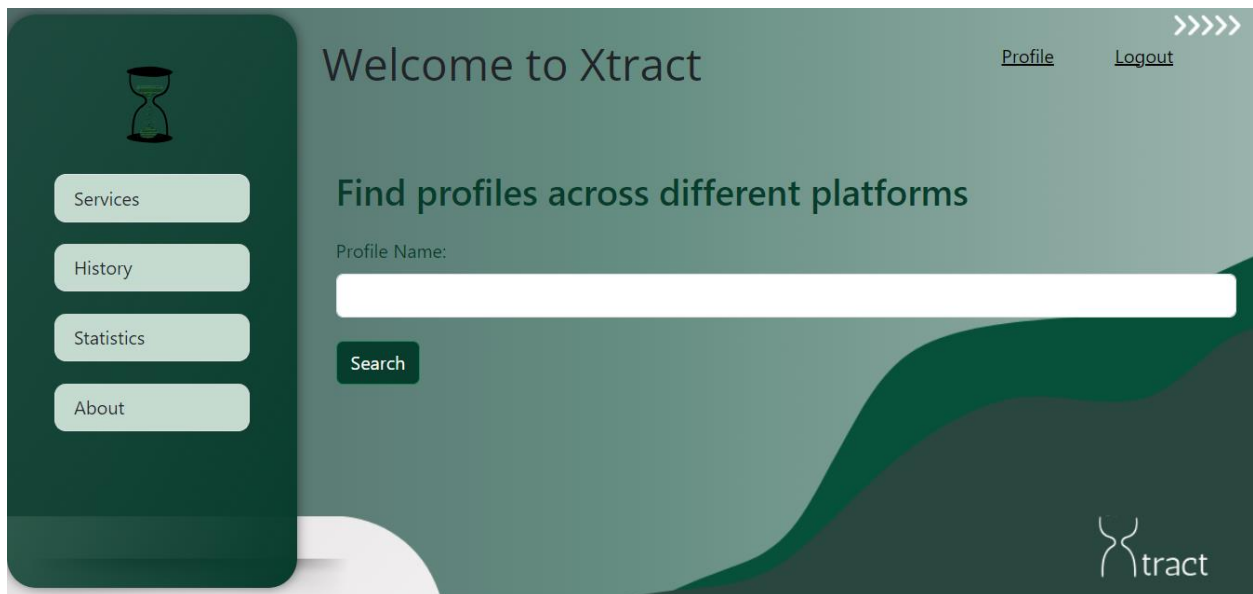


Figure 19 Cross platform GUI



Figure 20 About us page GUI

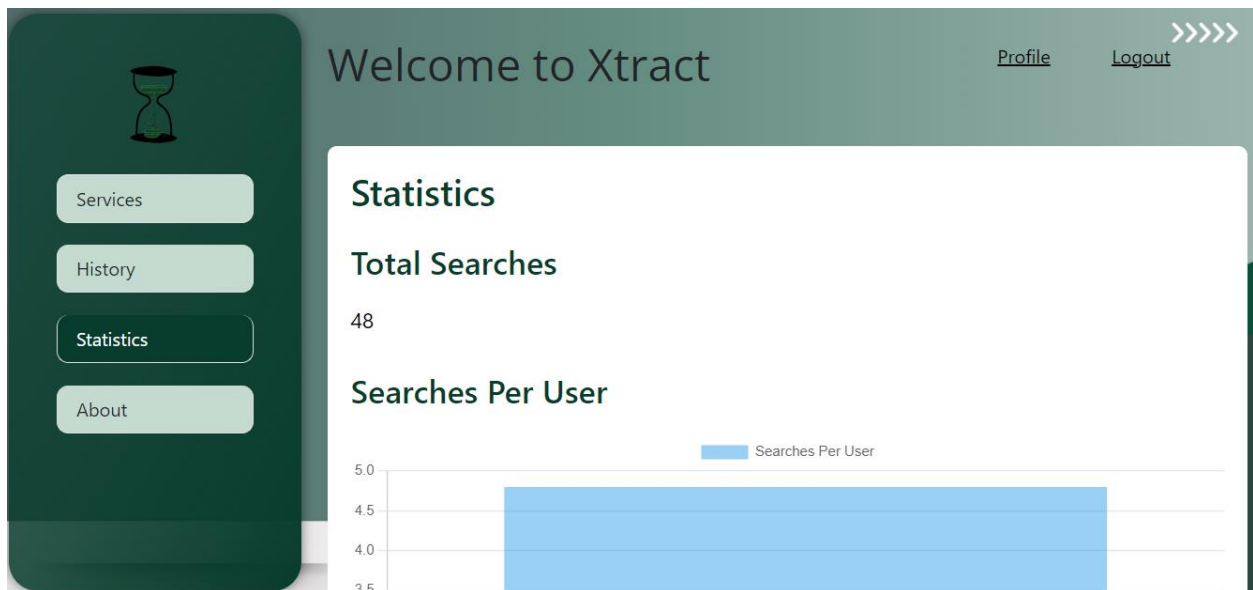


Figure 21 Statistics page GUI



Figure 22 History page GUI

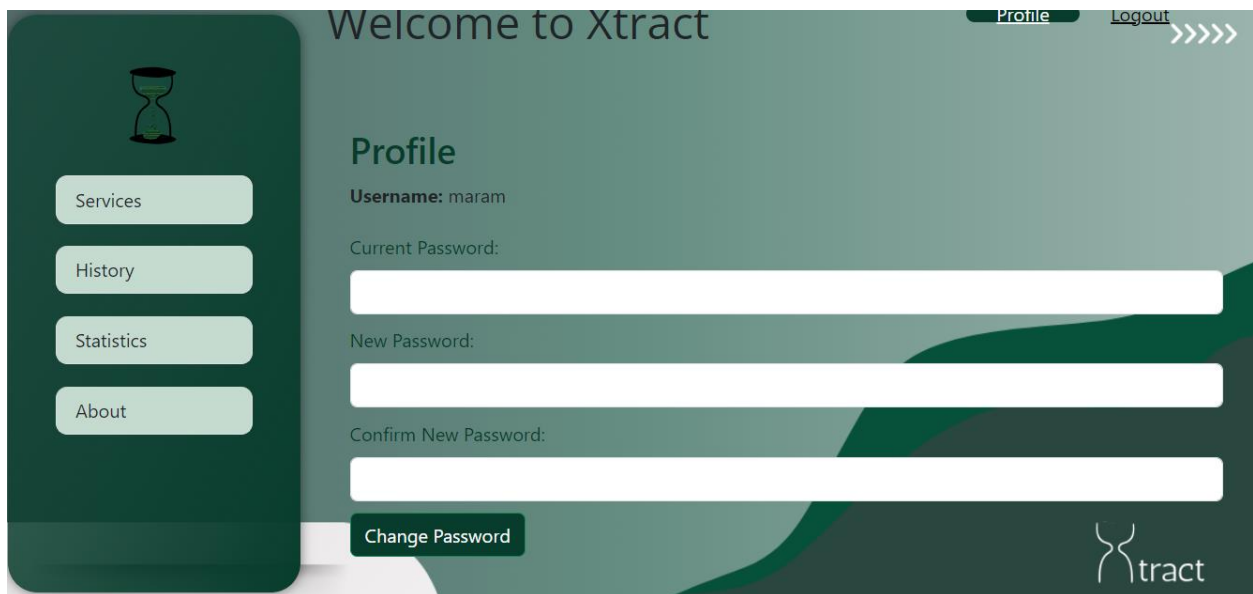



Figure 23 Profile page GUI

Welcome to Xtract [Profile](#) [Logout](#) >>>>




Services

History

Statistics

About



Search Method:


Search in user profile ▾

Enter LinkedIn Username:

Include: ☒ Name ☒ Subtitle ☒ About ☒ Experience ☒ Education ☒ Posts

Figure 24 LinkedIn search GUI

Welcome to Xtract [Profile](#) [Logout](#) >>>>




Services

History

Statistics

About




Search method:

Select option ▾

Extract

Figure 25 Facebook search GUI



Services

History

Statistics

About

Welcome to Xtract



Search Method:

Search in user profile

Username:

Keyword:

[Profile](#)[Logout](#)>>>>

Figure 26 Twitter search GUI

Chapter 5: Implementation and testing

5.1 Implementation

5.1.1 Frontend

We use Vue.js framework to build the frontend in our project, the main structure is the main App.vue file and the folders containing pages that build the project: Views, Components, Router, and assets as shown here in this file hierarchy:

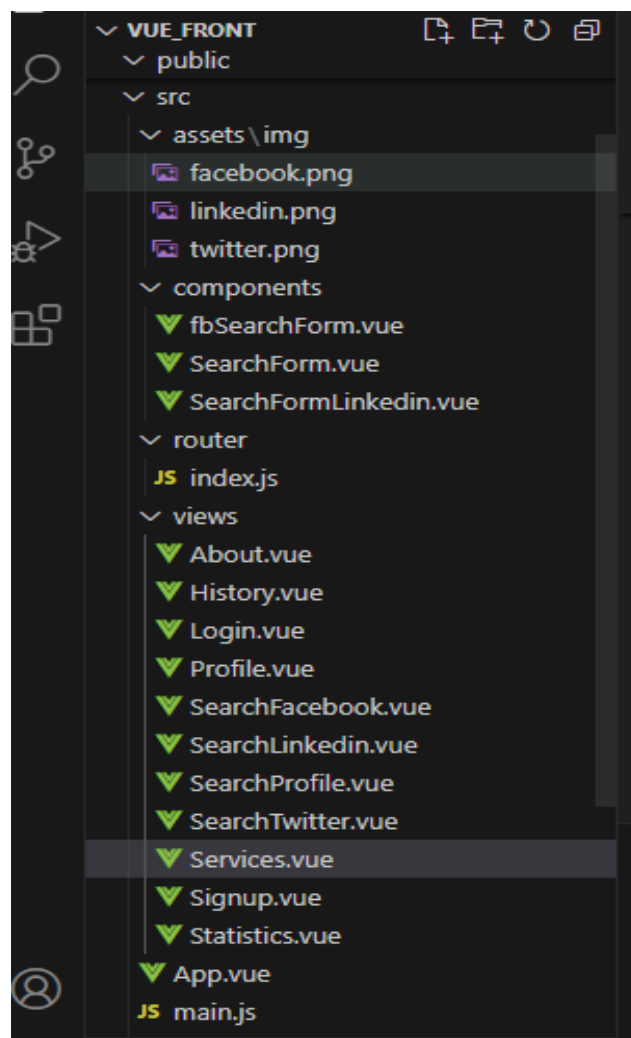
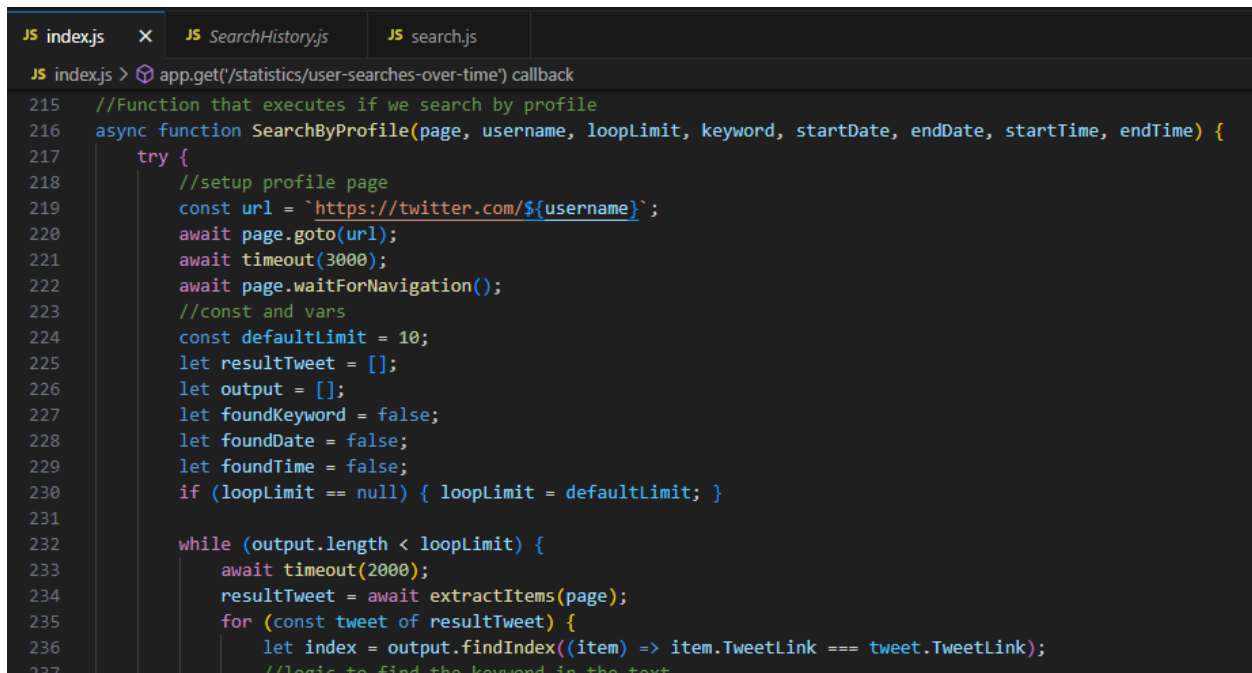


Figure 27 Frontend file hierarchy

5.1.2 Backend

Both Node.js and python are used to write the backend logic, we can say that there are total of three servers that contain the backend functionalities two of which are python servers running using the framework flask while the third is a Node.js server.

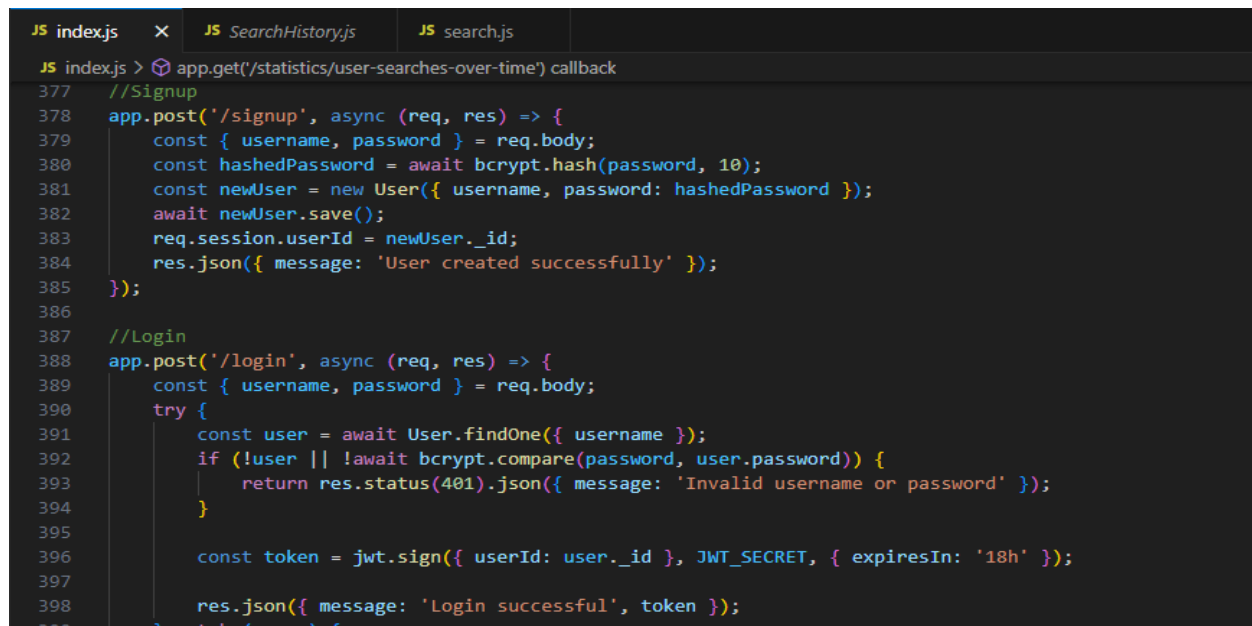
The node.js server basically hosts all the JavaScript code used for the service “Twitter search” as shown here in this code snippet:

A screenshot of a code editor with three tabs: 'index.js', 'SearchHistory.js', and 'search.js'. The 'search.js' tab is active, showing JavaScript code. The code starts with a comment 'app.get('/statistics/user-searches-over-time') callback' and a function 'SearchByProfile' that takes parameters: page, username, loopLimit, keyword, startDate, endDate, startTime, and endTime. The function uses 'async' and 'await' for asynchronous operations, including navigating to a Twitter profile, waiting for navigation, and extracting items. It also includes a 'while' loop to iterate through results and find a specific keyword. The code is line-numbered from 215 to 237.

```
JS index.js > app.get('/statistics/user-searches-over-time') callback
215 //Function that executes if we search by profile
216 async function SearchByProfile(page, username, loopLimit, keyword, startDate, endDate, startTime, endTime) {
217     try {
218         //setup profile page
219         const url = `https://twitter.com/${username}`;
220         await page.goto(url);
221         await timeout(3000);
222         await page.waitForNavigation();
223         //const and vars
224         const defaultLimit = 10;
225         let resultTweet = [];
226         let output = [];
227         let foundKeyword = false;
228         let foundDate = false;
229         let foundTime = false;
230         if (loopLimit == null) { loopLimit = defaultLimit; }
231
232         while (output.length < loopLimit) {
233             await timeout(2000);
234             resultTweet = await extractItems(page);
235             for (const tweet of resultTweet) {
236                 let index = output.findIndex((item) => item.TweetLink === tweet.TweetLink);
237                 //logic to find the keyword in the text
```

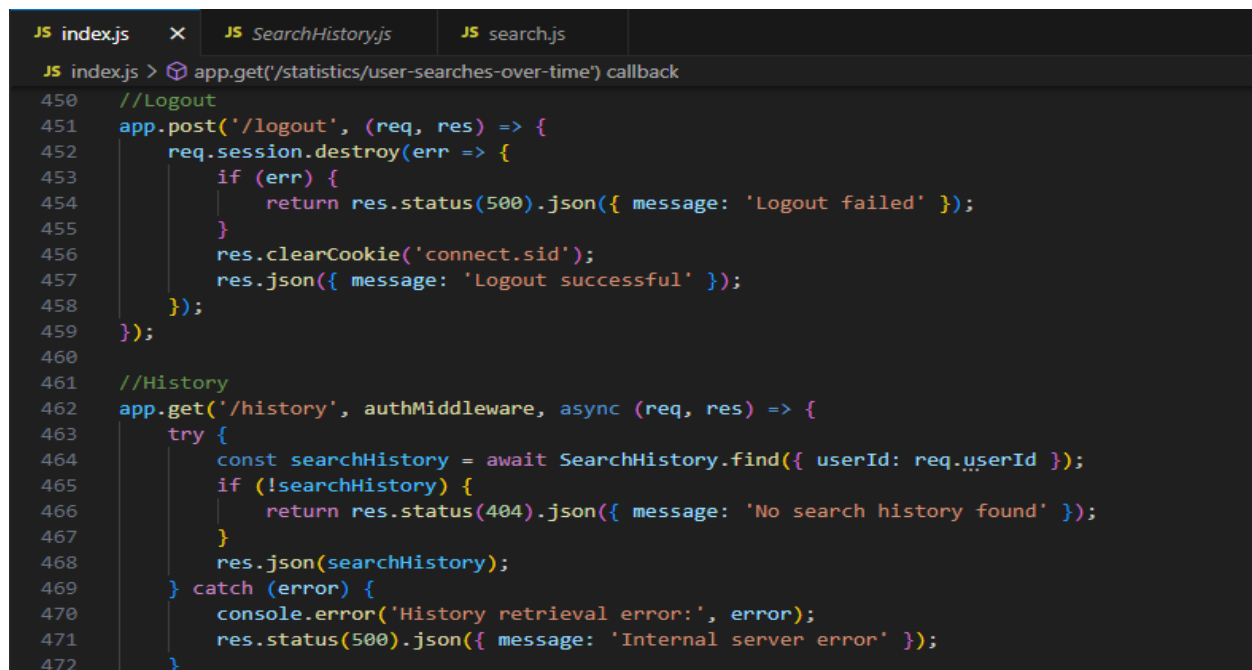
Figure 28 Node.js Backend ex. 1

not only that but it also has the functions responsible for handling the Signup, Login, profile, History, and Statistics pages, as shown here:



```
JS index.js x JS SearchHistory.js JS search.js
JS index.js > app.get('/statistics/user-searches-over-time') callback
377 //Signup
378 app.post('/signup', async (req, res) => {
379   const { username, password } = req.body;
380   const hashedPassword = await bcrypt.hash(password, 10);
381   const newUser = new User({ username, password: hashedPassword });
382   await newUser.save();
383   req.session.userId = newUser._id;
384   res.json({ message: 'User created successfully' });
385 });
386
387 //Login
388 app.post('/login', async (req, res) => {
389   const { username, password } = req.body;
390   try {
391     const user = await User.findOne({ username });
392     if (!user || !await bcrypt.compare(password, user.password)) {
393       return res.status(401).json({ message: 'Invalid username or password' });
394     }
395
396     const token = jwt.sign({ userId: user._id }, JWT_SECRET, { expiresIn: '18h' });
397
398     res.json({ message: 'Login successful', token });
399   } catch (error) {
```

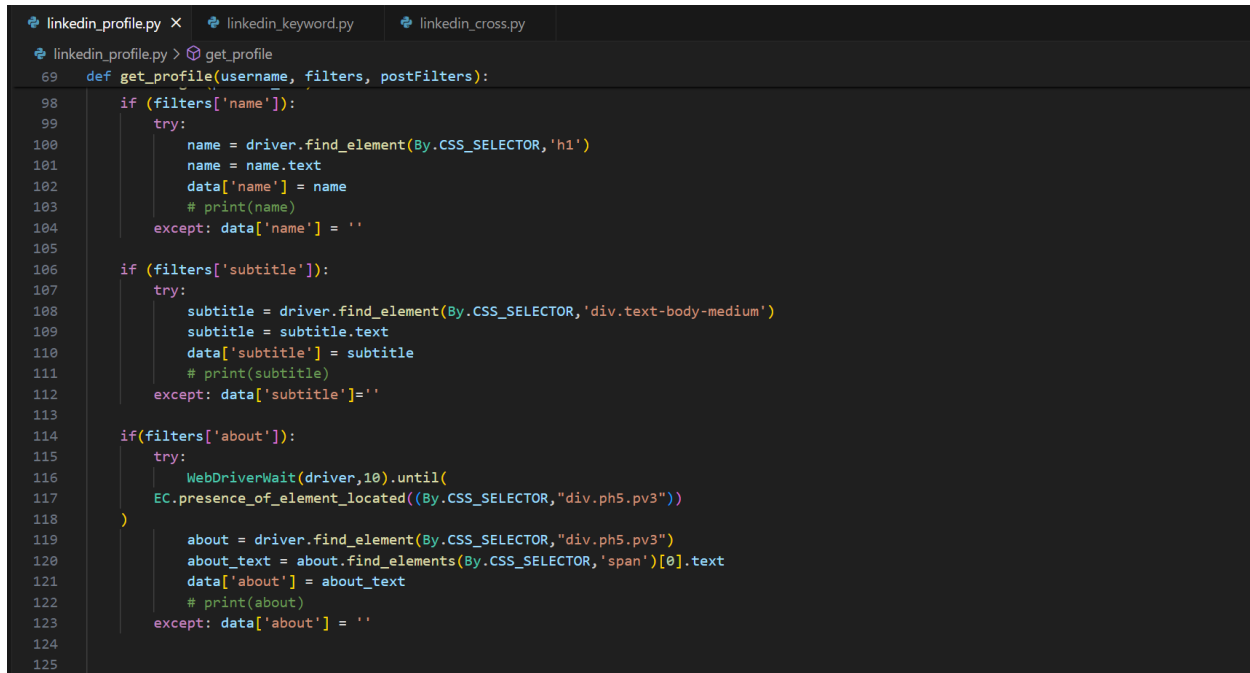
Figure 29 Node.js Backend ex.2



```
JS index.js x JS SearchHistory.js JS search.js
JS index.js > app.get('/statistics/user-searches-over-time') callback
450 //Logout
451 app.post('/logout', (req, res) => {
452   req.session.destroy(err => {
453     if (err) {
454       return res.status(500).json({ message: 'Logout failed' });
455     }
456     res.clearCookie('connect.sid');
457     res.json({ message: 'Logout successful' });
458   });
459 });
460
461 //History
462 app.get('/history', authMiddleware, async (req, res) => {
463   try {
464     const searchHistory = await SearchHistory.find({ userId: req.userId });
465     if (!searchHistory) {
466       return res.status(404).json({ message: 'No search history found' });
467     }
468     res.json(searchHistory);
469   } catch (error) {
470     console.error('History retrieval error:', error);
471     res.status(500).json({ message: 'Internal server error' });
472   }
473 }
```

Figure 30 Node.js Backend ex.3

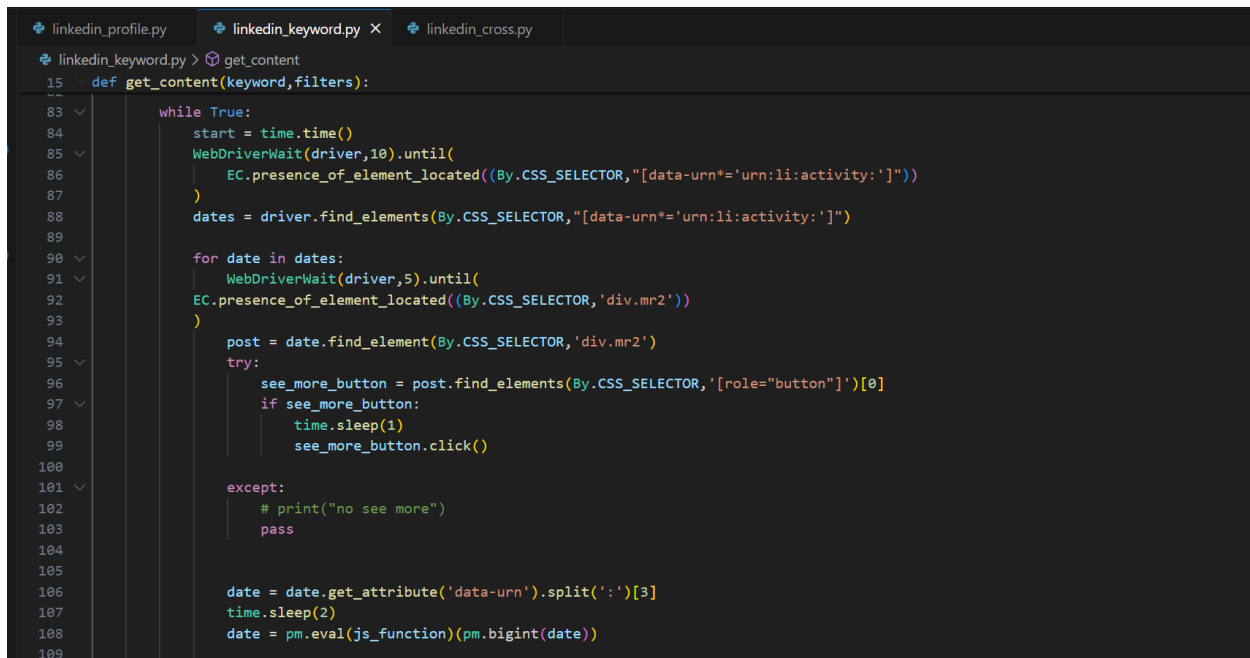
The first python server houses the logic for all the scraping needed in the “LinkedIn Service” part of the app and the code needed to accept requests from the front end and respond to them. The logic is divided across three files. One for scraping a user’s profile.



```
linkedin_profile.py X linkedin_keyword.py linkedin_cross.py
linkedin_profile.py > get_profile
69 def get_profile(username, filters, postFilters):
98     if (filters['name']):
99         try:
100             name = driver.find_element(By.CSS_SELECTOR, 'h1')
101             name = name.text
102             data['name'] = name
103             # print(name)
104         except: data['name'] = ''
105
106     if (filters['subtitle']):
107         try:
108             subtitle = driver.find_element(By.CSS_SELECTOR, 'div.text-body-medium')
109             subtitle = subtitle.text
110             data['subtitle'] = subtitle
111             # print(subtitle)
112         except: data['subtitle'] = ''
113
114     if(filters['about']):
115         try:
116             WebDriverWait(driver,10).until(
117                 EC.presence_of_element_located((By.CSS_SELECTOR, "div.ph5.pv3"))
118             )
119             about = driver.find_element(By.CSS_SELECTOR, "div.ph5.pv3")
120             about_text = about.find_elements(By.CSS_SELECTOR, 'span')[0].text
121             data['about'] = about_text
122             # print(about)
123         except: data['about'] = ''
124
125
```

Figure 31 Python Backend ex.1

The second file contains the code for scraping posts using a specific keyword as well as filtering the results further by date and amount.



```
15 def get_content(keyword, filters):
16
17     while True:
18         start = time.time()
19         WebDriverWait(driver, 10).until(
20             EC.presence_of_element_located((By.CSS_SELECTOR, "[data-urn*='urn:li:activity:']"))
21         )
22         dates = driver.find_elements(By.CSS_SELECTOR, "[data-urn*='urn:li:activity:']")
23
24         for date in dates:
25             WebDriverWait(driver, 5).until(
26                 EC.presence_of_element_located((By.CSS_SELECTOR, 'div.mr2'))
27             )
28             post = date.find_element(By.CSS_SELECTOR, 'div.mr2')
29             try:
30                 see_more_button = post.find_elements(By.CSS_SELECTOR, '[role="button"]')[0]
31                 if see_more_button:
32                     time.sleep(1)
33                     see_more_button.click()
34
35             except:
36                 # print("no see more")
37                 pass
38
39             date = date.get_attribute('data-urn').split(':')[3]
40             time.sleep(2)
41             date = pm.eval(js_function)(pm.bigint(date))
```

Figure 32 Python Backend ex.2

The third file is part of the “cross-platform search: functionality. It has a code that searches for profiles in LinkedIn.

```
linkedin_cross.py > find_profile
11 def find_profile(name):
66     list = driver.find_element(By.CSS_SELECTOR,"div.mb2")
67
68     profile_list = list.find_elements(By.CSS_SELECTOR,"li")
69
70     for profile in profile_list:
71         try:
72             profile_pic=profile.find_element(By.CSS_SELECTOR,'img').get_attribute('src')
73         except: profile_pic=None
74         try:
75             profile_name= profile.find_element(By.CSS_SELECTOR,'span[dir="ltr"]').find_element(By.CSS_SELECTOR,'span').text
76         except: profile_name=""
77         try:
78             profile_url= profile.find_element(By.CSS_SELECTOR,'a').get_attribute('href')
79         except: profile_url=''
80
81
82     if not[any(profile['ProfileLink'] == profile_url for profile in profiles)]:
83         profiles.append({
84             'ProfileImg':profile_pic,
85             'ProfileName':profile_name,
86             'ProfileLink':profile_url
87         })
88
89     print(profiles)
90     if (len(profiles) >= max_results):
91         break
92
93
```

Figure 33 Python Backend ex.3

The “app.py” file has the routes that the front end sends requests to and the connection to the database.

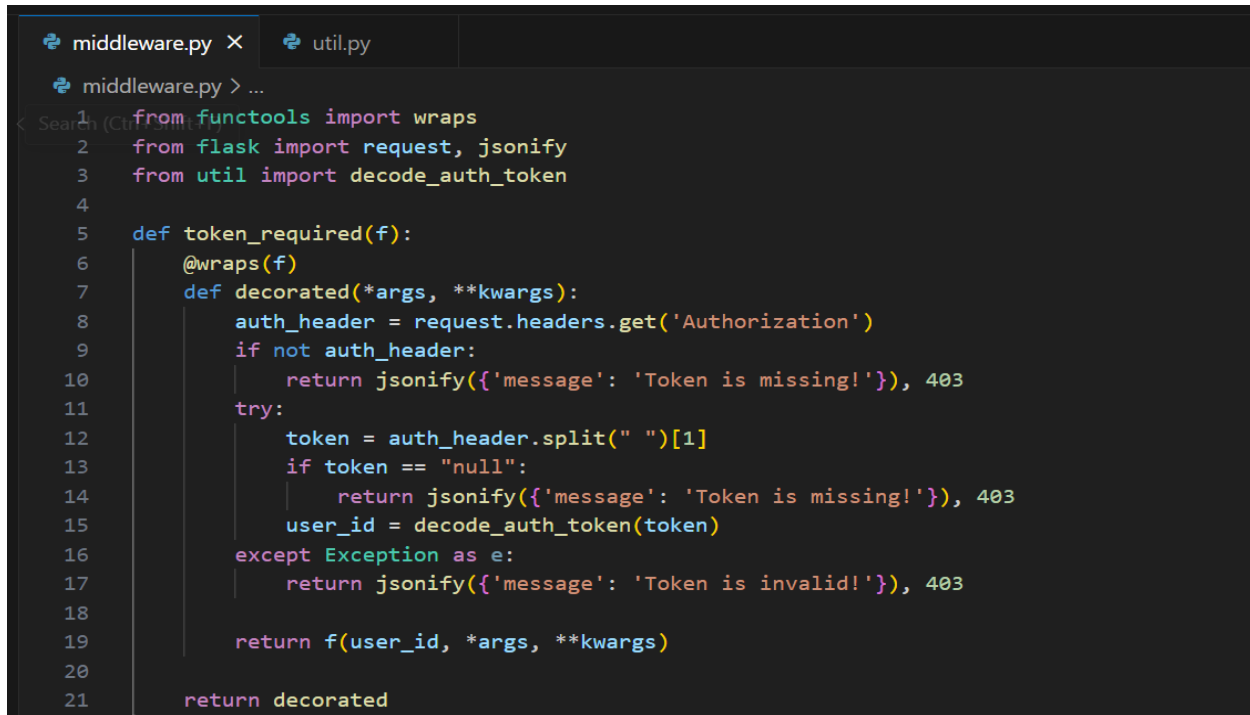
```
app.py M X
app.py > ...
44
45 @app.route('/searchLinkedin',methods=['POST'])
46 @token_required
47 def search_linkedin(user_id):
48     print(user_id)
49     print(request.json)
50     data = {}
51     if request.method == 'POST':
52         data['searchMethod'] = request.json['searchMethod']
53         data['username'] = request.json['username']
54         data['keyword'] = request.json['keyword']
55         data['loopLimit'] = request.json['loopLimit']
56         data['startDate'] = request.json['startDate']
57         data['endDate'] = request.json['endDate']
58         data['startTime'] = request.json['startTime']
59         data['endTime'] = request.json['endTime']
60         data['filters'] = request.json['filters']
61
62         utc3 = pytz.timezone('Etc/GMT+3')
63         now_utc3 = datetime.datetime.now(utc3)
64         # Format the datetime
65         formatted_date = now_utc3.strftime('%Y-%m-%d %H:%M')
66
67         search_history = {
68             "service": "Linkedin",
69             "query": data["username"] or data["keyword"],
70             "date": formatted_date,
71             "userId": ObjectId(user_id)
72         }
```

Figure 34 Python Backend ex.4

```
app.py M X
app.py > ...
125
126
127
128 @app.route('/download/<file_name>',methods=['GET'])
129 def download(file_name):
130     format = request.args.get('fileFormat')
131     if format == 'csv':
132         file_path = f"./download/{file_name}.csv"
133         return send_file(file_path,download_name=file_name,as_attachment=True)
134     elif format == 'json':
135         file_path = f"./download/{file_name}.json"
136         return send_file(file_path,download_name=file_name,as_attachment=True)
137
138     return jsonify('Not found 404')
139
140
141 @app.route('/findProfile',methods=['POST'])
142 @token_required
143 def search(user_id):
144     name = request.json['profileName']
145
146     profiles = find_profile(name)
147     # print(profiles)
148     return jsonify (profiles = profiles)
149
150
151 #
152 if __name__ == '__main__':
153     app.run(port=5000,debug=True)
```

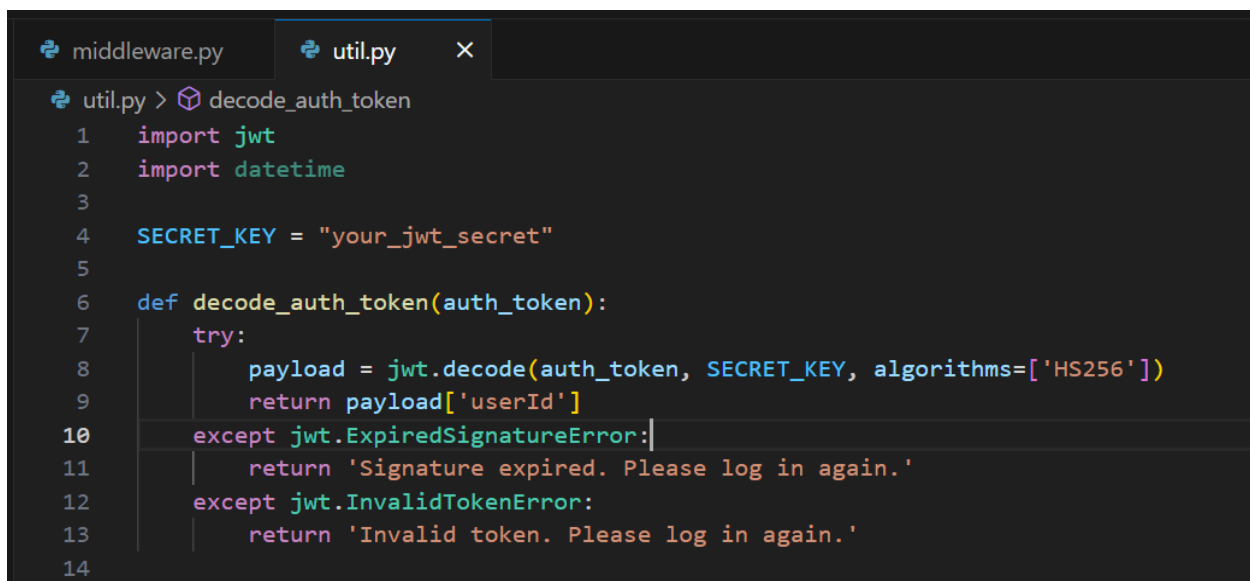
Figure 35 Python Backend ex.5

The server also has two files that retrieve and decode the user's token to verify his identity. The server doesn't accept any request with a token that fails the decoding.



```
1 from functools import wraps
2 from flask import request, jsonify
3 from util import decode_auth_token
4
5 def token_required(f):
6     @wraps(f)
7     def decorated(*args, **kwargs):
8         auth_header = request.headers.get('Authorization')
9         if not auth_header:
10             return jsonify({'message': 'Token is missing!'}), 403
11         try:
12             token = auth_header.split(" ")[1]
13             if token == "null":
14                 return jsonify({'message': 'Token is missing!'}), 403
15             user_id = decode_auth_token(token)
16         except Exception as e:
17             return jsonify({'message': 'Token is invalid!'}), 403
18
19         return f(user_id, *args, **kwargs)
20
21     return decorated
```

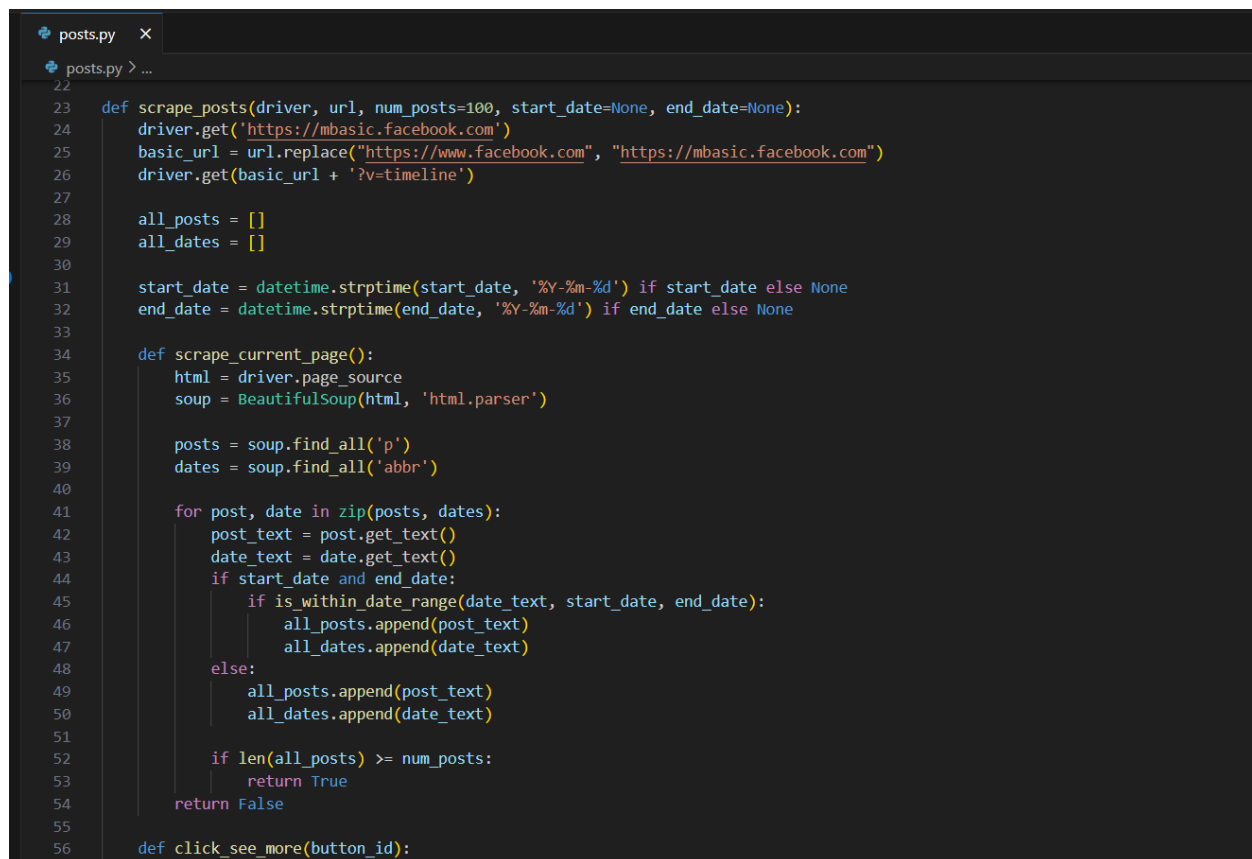
Figure 36 Python Backend ex.6



```
1 import jwt
2 import datetime
3
4 SECRET_KEY = "your_jwt_secret"
5
6 def decode_auth_token(auth_token):
7     try:
8         payload = jwt.decode(auth_token, SECRET_KEY, algorithms=['HS256'])
9         return payload['userId']
10    except jwt.ExpiredSignatureError:
11        return 'Signature expired. Please log in again.'
12    except jwt.InvalidTokenError:
13        return 'Invalid token. Please log in again.'
14
```

Figure 37 Python Backend ex.7

The second python server houses the logic for all the scraping needed in the “Facebook Service” part of the app and the code needed to accept requests from the front end and respond to them. The logic is divided across six files. The first file for scraping user posts with filtering the result by number of posts and date.



```
posts.py x
posts.py > ...
22
23 def scrape_posts(driver, url, num_posts=100, start_date=None, end_date=None):
24     driver.get('https://mbasic.facebook.com')
25     basic_url = url.replace("https://www.facebook.com", "https://mbasic.facebook.com")
26     driver.get(basic_url + '?v=timeline')
27
28     all_posts = []
29     all_dates = []
30
31     start_date = datetime.strptime(start_date, '%Y-%m-%d') if start_date else None
32     end_date = datetime.strptime(end_date, '%Y-%m-%d') if end_date else None
33
34     def scrape_current_page():
35         html = driver.page_source
36         soup = BeautifulSoup(html, 'html.parser')
37
38         posts = soup.find_all('p')
39         dates = soup.find_all('abbr')
40
41         for post, date in zip(posts, dates):
42             post_text = post.get_text()
43             date_text = date.get_text()
44             if start_date and end_date:
45                 if is_within_date_range(date_text, start_date, end_date):
46                     all_posts.append(post_text)
47                     all_dates.append(date_text)
48             else:
49                 all_posts.append(post_text)
50                 all_dates.append(date_text)
51
52             if len(all_posts) >= num_posts:
53                 return True
54         return False
55
56     def click_see_more(button_id):
```

Figure 38 Python Backend ex.8

The second file contains the code for scraping user information such as education, work, living place, contact info, relationship ...etc

```
information.py x
information.py > profile_info > driver
11
12 def profile_info(driver, url):
13     driver.get('https://mbasic.facebook.com')
14     basic_url = url.replace("https://www.facebook.com", "https://mbasic.facebook.com")
15     driver.get(basic_url)
16
17     html = driver.page_source
18     soup = BeautifulSoup(html, 'html.parser')
19
20     profile_data = {}
21
22     sections = {
23         "Education": 'education',
24         "Work": 'work',
25         "Living": 'living',
26         "Contact Info": 'contact-info',
27         "Basic Info": 'basic-info',
28         "Relationship": 'relationship',
29         "Bio": 'bio',
30         "Events": 'events'
31     }
32
33     for section_name, section_id in sections.items():
34         try:
35             section = soup.find('div', id=section_id)
36             section_texts = extract_section_text(section)
37             profile_data[section_name] = section_texts
38         except:
39             pass
40
41     return profile_data
42
43 def save_to_csv(profile_data, filename='profile_info.csv'):
44     with open(filename, 'w', newline='', encoding='utf-8-sig') as csvfile:
45         writer = csv.writer(csvfile)
46         writer.writerow(['Section', 'Information'])
47         for section, texts in profile_data.items():
```

Figure 39 Python Backend ex.9

The third file contains the code for scraping user photos.

```
userPhotos.py x
userPhotos.py > ...
1 from selenium.webdriver.common.by import By
2 import time
3 import json
4 from login import login_facebook
5 import pandas as pd
6
7 # function to get user photo
8 def user_photos(driver, url):
9     all_photos = []
10    driver.get(url)
11    for i in ["photos_of", "photos_by"]:
12        driver.get(url + "/" + i)
13        time.sleep(25)
14        SCROLL_PAUSE_TIME = 5
15        driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
16        time.sleep(SCROLL_PAUSE_TIME)
17        anchors = driver.find_elements(By.CSS_SELECTOR, "[class='x9f619 x1r8uery x1iyjqo2 x6ikm8r x10wlt62 x1n2onr6']")
18        time.sleep(5)
19        for img in anchors:
20            try:
21                user_images = img.find_element(By.TAG_NAME, "img")
22                image_link = user_images.get_attribute('src')
23            except:
24                pass
25            all_photos.append(image_link)
26
27    for photo in all_photos:
28        print(f"photo link: {photo}\n")
29
30    return all_photos
```

Figure 40 Python Backend ex.10

The fourth file contains the code for scraping generic posts using a specific keyword.

```
searchKeyword.py X
searchKeyword.py > searchText
7 def scroll(driver):
14     time.sleep(SCROLL_PAUSE_TIME)
15     new_height = driver.execute_script("return document.body.scrollHeight")
16     if new_height == page_height:
17         break
18     page_height = new_height
19     counter = counter + 1
20
21
22 def searchText(driver, textToSearch):
23     all_text = []
24     all_link = []
25     search_box = driver.find_element(By.XPATH, "//*[@type='search']")
26     search_box.clear()
27     search_box.send_keys(textToSearch)
28     time.sleep(2)
29     search_box.send_keys(Keys.ENTER)
30     time.sleep(5)
31     print(search_box)
32     search_box.clear()
33
34     driver.get(f"https://www.facebook.com/search/posts/?q={textToSearch}")
35     scroll(driver)
36
37     posts = driver.find_elements(By.CSS_SELECTOR, "[class='x1n2onr6 x1ja2u2z']")
38     for post in posts:
39         try:
40             txt = post.find_element(By.CSS_SELECTOR, "[class='xdj266r x11i5rnm xat24cr x1mh8g0r x1vvkbs x126k92a']")
41             txt_t = txt.text
42             print(txt_t)
43         except:
44             txt_t = None
45             all_text.append(txt_t)
46     for link in posts:
47         try:
48             lnk = link.find_element(By.TAG_NAME, "a")
```

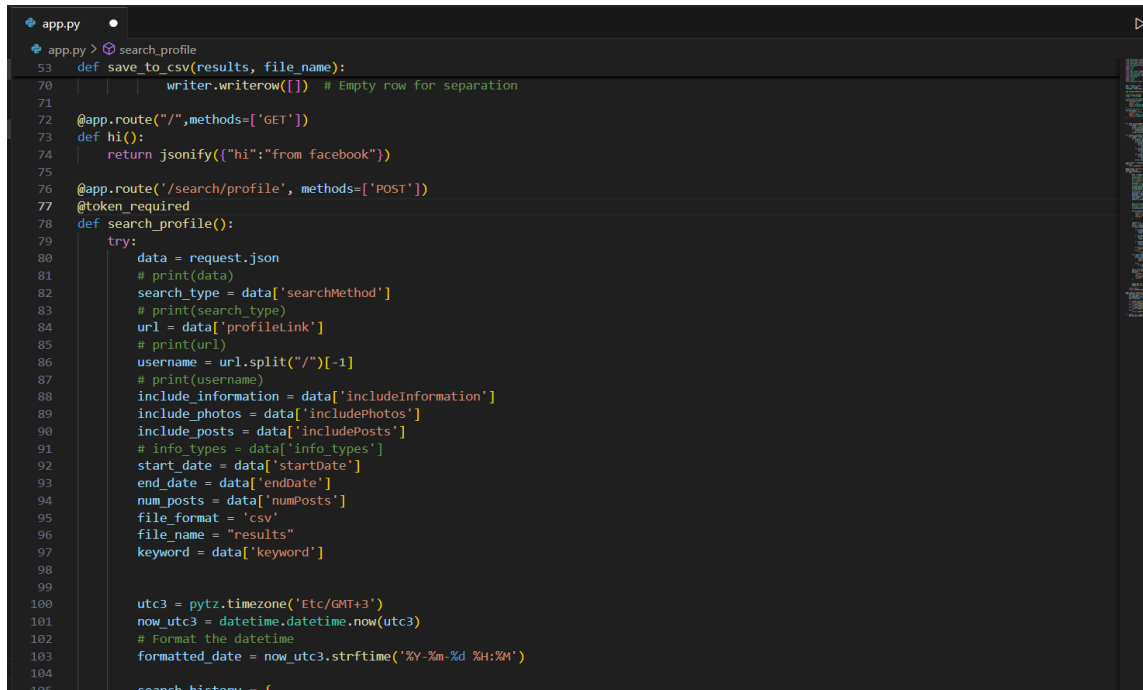
Figure 41 Python Backend ex.11

The fifth file is part of the “cross-platform search: functionality. It has a code that searches for profiles in Facebook.

```
search.py X
search.py > ...
30 def search_by_name(nameToSearch):
31     #search by name
32     search_box = driver.find_element(By.XPATH, "//*[@type='search']")
33     search_box.clear()
34     search_box.send_keys(nameToSearch)
35     time.sleep(2)
36     search_box.send_keys(Keys.ENTER)
37     time.sleep(5)
38     print(search_box)
39     search_box.clear()
40
41
42     #see all
43     see_all = driver.find_element(By.CSS_SELECTOR, "[aria-label='See all']")
44     if see_all:
45         see_all.click()
46
47     #scroll down with users name and password
48
49     SCROLL_PAUSE_TIME = 2
50     page_height = 0
51     while True:
52         driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
53         time.sleep(SCROLL_PAUSE_TIME)
54         new_height = driver.execute_script("return document.body.scrollHeight")
55         if new_height == page_height:
56             break
57         page_height = new_height
58
59     #user profile and image and description
60     all_photos = []
61     all_links = []
62
63     searchResults = driver.find_elements(By.CSS_SELECTOR, "[class='x1yztbdb']")
64     for img in searchResults:
65         try:
66             users_profilePhoto = img.find_element(By.TAG_NAME, "img")
```

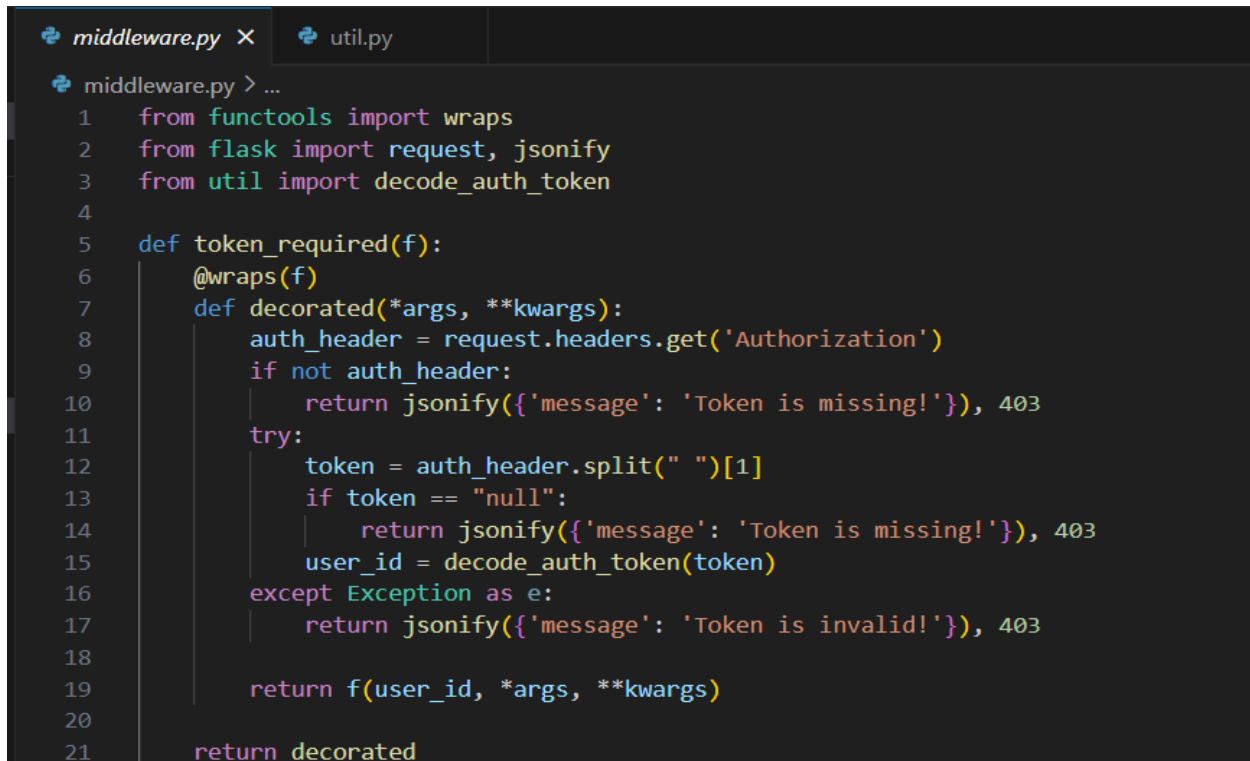
Figure 42 Python Backend ex.12

The “app.py” file has the routes that the front end sends requests to and the connection to the database.



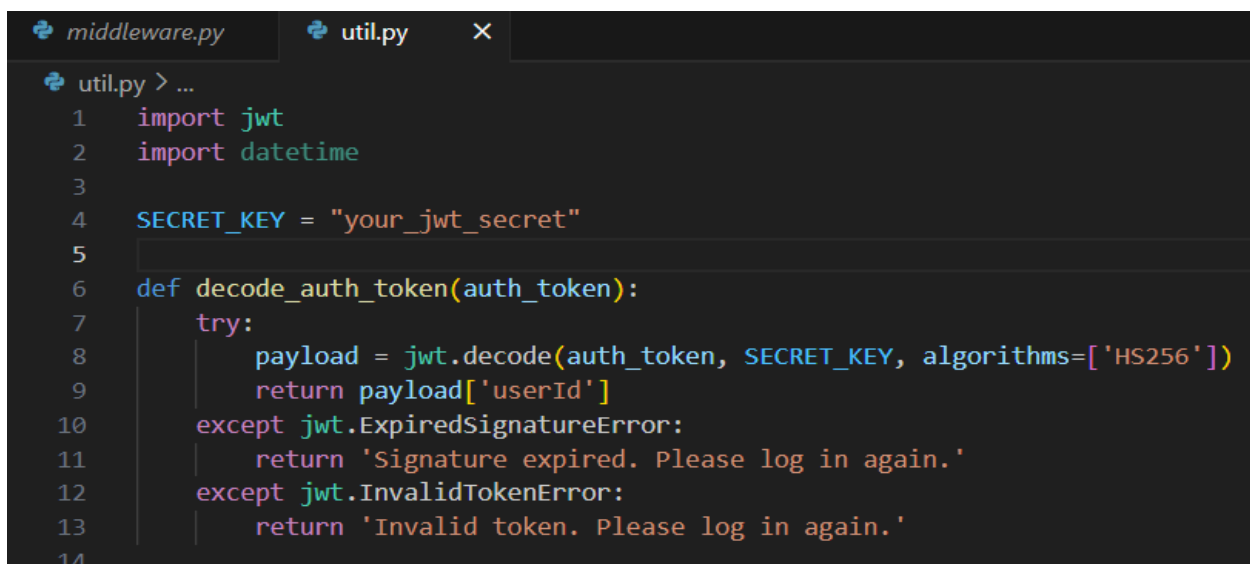
```
app.py
app.py > search_profile
53 def save_to_csv(results, file_name):
54     writer.writerow([]) # Empty row for separation
55
56
57
58
59
60
61
62 @app.route("/", methods=['GET'])
63 def hi():
64     return jsonify({"hi": "from facebook"})
65
66
67
68 @app.route('/search/profile', methods=['POST'])
69 @token_required
70 def search_profile():
71     try:
72         data = request.json
73         # print(data)
74         search_type = data['searchMethod']
75         # print(search_type)
76         url = data['profileLink']
77         # print(url)
78         username = url.split("/")[-1]
79         # print(username)
80         include_information = data['includeInformation']
81         include_photos = data['includePhotos']
82         include_posts = data['includePosts']
83         # info_types = data['info_types']
84         start_date = data['startDate']
85         end_date = data['endDate']
86         num_posts = data['numPosts']
87         file_format = 'csv'
88         file_name = "results"
89         keyword = data['keyword']
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
```

The server also has two files that retrieve and decode the user's token to verify his identity. The server doesn't accept any request with a token that fails the decoding.



```
middleware.py X util.py
middleware.py > ...
1  from functools import wraps
2  from flask import request, jsonify
3  from util import decode_auth_token
4
5  def token_required(f):
6      @wraps(f)
7      def decorated(*args, **kwargs):
8          auth_header = request.headers.get('Authorization')
9          if not auth_header:
10             return jsonify({'message': 'Token is missing!'}), 403
11         try:
12             token = auth_header.split(" ")[1]
13             if token == "null":
14                 return jsonify({'message': 'Token is missing!'}), 403
15             user_id = decode_auth_token(token)
16         except Exception as e:
17             return jsonify({'message': 'Token is invalid!'}), 403
18
19         return f(user_id, *args, **kwargs)
20
21     return decorated
```

Figure 45 Python Backend ex.15



```
middleware.py util.py X
util.py > ...
1  import jwt
2  import datetime
3
4  SECRET_KEY = "your_jwt_secret"
5
6  def decode_auth_token(auth_token):
7      try:
8          payload = jwt.decode(auth_token, SECRET_KEY, algorithms=['HS256'])
9          return payload['userId']
10     except jwt.ExpiredSignatureError:
11         return 'Signature expired. Please log in again.'
12     except jwt.InvalidTokenError:
13         return 'Invalid token. Please log in again.'
14
```

Figure 46 Python Backend ex.16

5.1.3 Database

We use the cloud service that MongoDB provides to save our data in three main collections which are, users, search, and searchhistories as shown here:

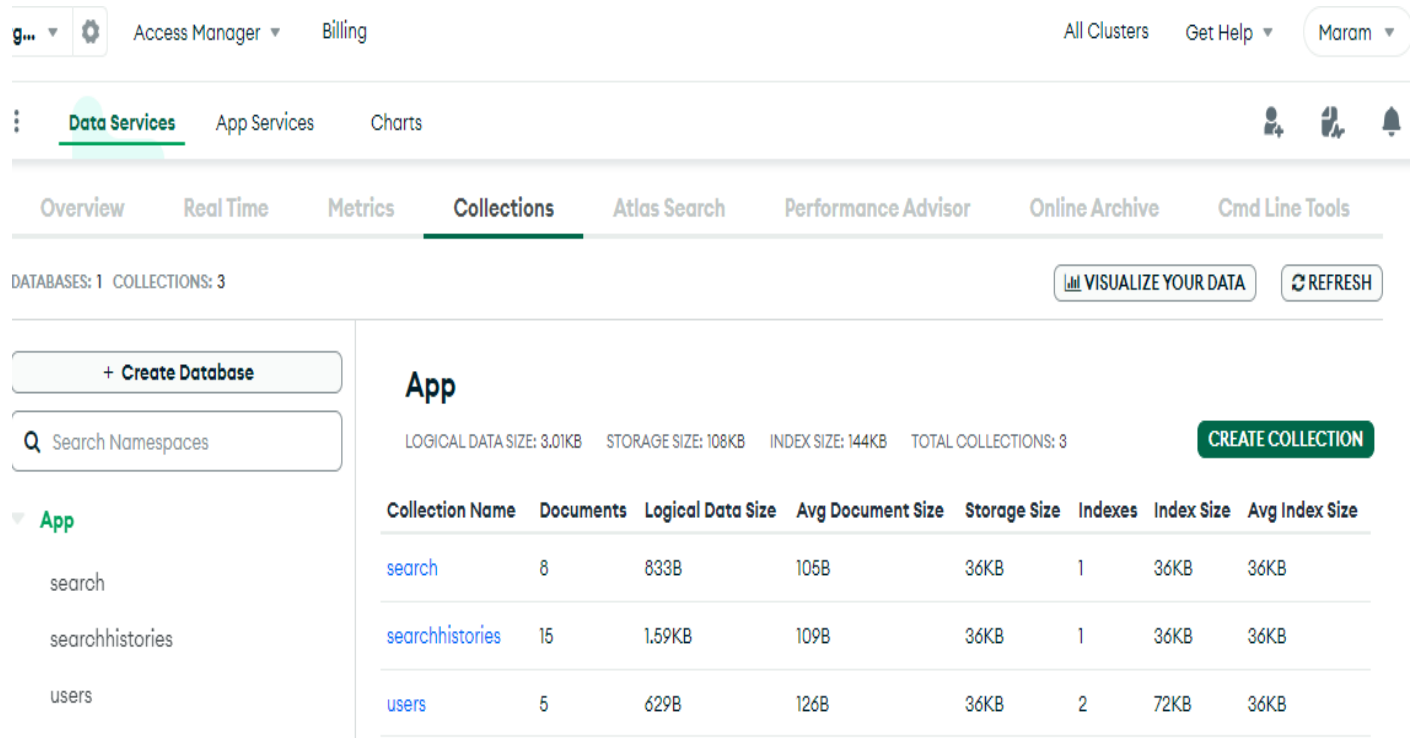


Figure 47 Cloud database collections

5.2 Testing

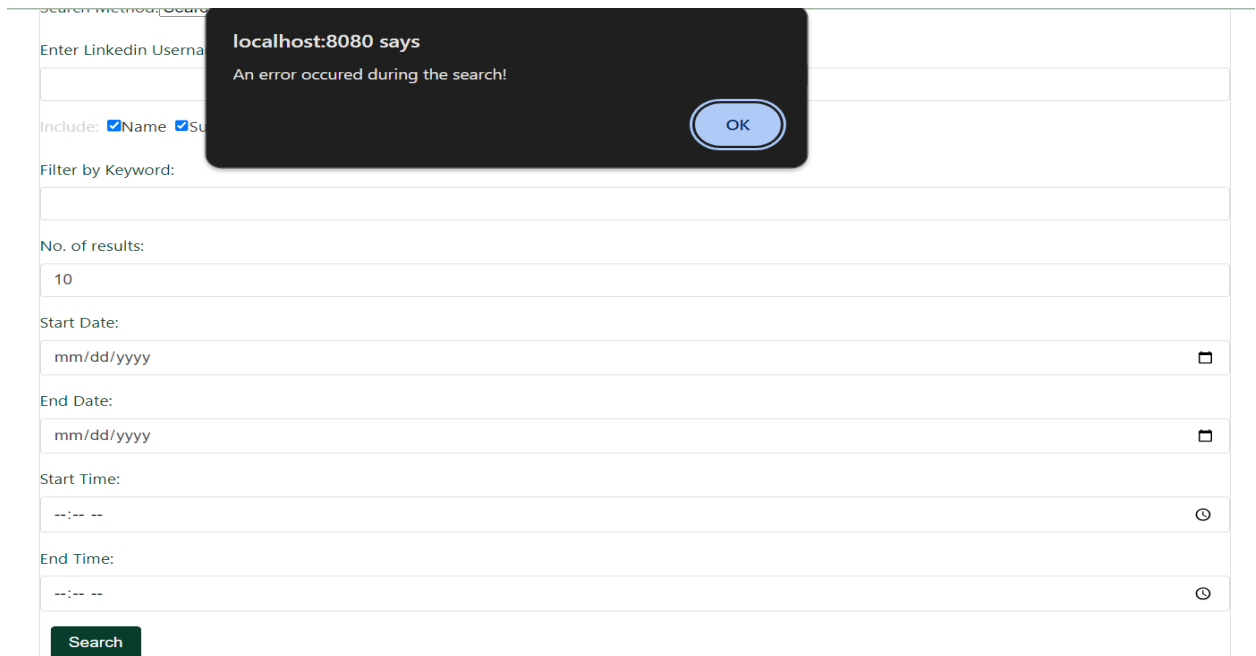
Functionality	Input	Expected Output
Sign up	Empty	A prompt tells the user to fill the username and password fields.
	Already used username with used or unused password.	A prompt tells user that sign up failed.
	Already used password with unused username.	Sign up is successful.
	New unused username and password.	Sign up successful and directed to Profile page.
Log in	Empty	A prompt tells the user to fill the username and password fields.
	Wrong username or password.	Log in failed alert.
	Valid username and password.	Log in successful and directed to profile page.
Twitter Search	No provided twitter username or keyword.	User cannot search unless both are provided.
	Enter loop limit of 0 or less or decimal.	A prompt tells the user to enter number that is equal to 1 or greater.
	Only Twitter username and keyword without providing data and time.	The search is done successfully.
Linkedin Search	A user tries to search without providing a username.	The app pop up "An error occurred during the search!". The user needs to enter at least a username to carry on with the search.

	A user enters a username but doesn't select any data to be retrieved.	The app accepts the request and runs the code, but the results come back empty.
	The user asks the app to retrieve a section that is not there on the searched user account.	The app accepts request, and the search is done with an empty response.
	A user selects one or more sections for the search.	The search is done, and values of the attributes are returned if found.
	A user tries to search when his token is expired.	The app tells the user an error occurred and wait for the user to login again.
Facebook Search	A user tries to search without providing a URL.	The app pop up "An error occurred during the search!". The user needs to enter URL to continue
	A user enters a username but doesn't select any data to be retrieved.	The app accepts the request and runs the code, but the results come back empty.
	A user selects one or more sections for the search.	The search is done, and values of the attributes are returned if found.

Table 2 Testing

LinkedIn Test Cases:

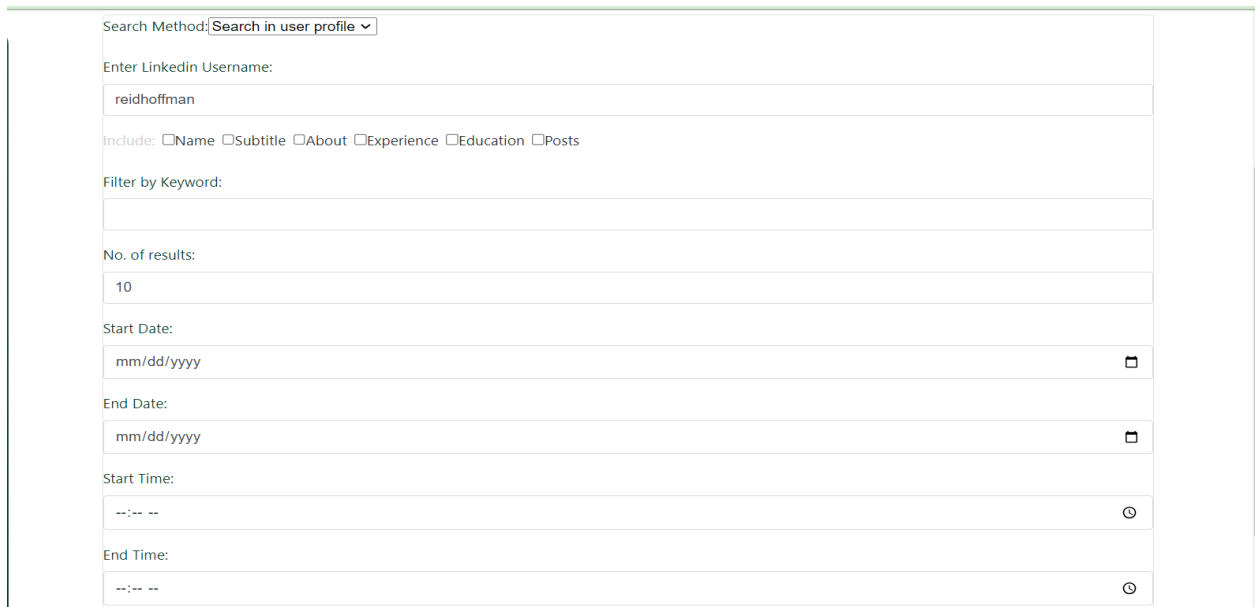
- 1) A user tries to search without providing a username:



The screenshot shows the LinkedIn search interface. At the top, there is a search bar with the placeholder text "Search Method: Search". Below it, the "Enter LinkedIn Username" field is empty. To the right of this field, a dark blue error message box is displayed, stating "localhost:8080 says" and "An error occurred during the search!". Below the error box, the "Include:" section shows "Name" and "Surname" selected with blue checkmarks. The "Filter by Keyword:" field is empty. Below this, the "No. of results:" field shows "10". The "Start Date:" field shows "mm/dd/yyyy" with a calendar icon. The "End Date:" field shows "mm/dd/yyyy" with a calendar icon. The "Start Time:" field shows "--:-- --" with a clock icon. The "End Time:" field shows "--:-- --" with a clock icon. At the bottom, there is a green "Search" button.

Figure 48 Test case 1

- 2) A user enters a username but doesn't select any data to be retrieved:



The screenshot shows the LinkedIn search interface. At the top, there is a search bar with the placeholder text "Search Method: Search in user profile". Below it, the "Enter LinkedIn Username:" field contains the text "reidhoffman". To the right of this field, the "Include:" section shows "Name", "Subtitle", "About", "Experience", "Education", and "Posts" all with unchecked checkboxes. The "Filter by Keyword:" field is empty. Below this, the "No. of results:" field shows "10". The "Start Date:" field shows "mm/dd/yyyy" with a calendar icon. The "End Date:" field shows "mm/dd/yyyy" with a calendar icon. The "Start Time:" field shows "--:-- --" with a clock icon. The "End Time:" field shows "--:-- --" with a clock icon.

Figure 49 Test case 2.1

Search

Results File Name:

Status:

File Format: CSV

Download

Success

RESULTS

Figure 50 Test case 2.2

3) A user selects one or more sections for the search:

Welcome to Xtract

ProfileLogout

in

LinkedIn

Search Method: Search in user profile

Enter LinkedIn Username:

reidhoffman

Include: ☒Name ☒Subtitle ☒About ☒Experience ☒Education ☒Posts

Filter by Keyword:

No. of results:

5

Start Date:

mm/dd/yyyy

End Date:

Figure 51 Test case 3.1

Results File Name:

Status:

File Format:JSON

Download

reidhoffman-profile-info

Success

reidhoffman-posts

RESULTS

Figure 52 Test case 3.2

RESULTS

Profile Info:

Name:

Reid Hoffman

Subtitle:

Co-Founder, LinkedIn & Inflection AI. Investor at Greylock.

About:

My two current priorities are: (1) using AI to benefit humanity and (2) protecting US democracy. I am active in all facets of the consumer internet and software industries. My focus spans product development, innovation, business strategy, and finance. My expertise also extends to general management, operations, business development, talent management, and marketing. I have experience with seed-stage companies such as PayPal, LinkedIn, Facebook, Zynga, Last.fm, and Flickr, as well as growth companies like Mozilla, LinkedIn, Zynga, and PayPal. My specialties include general management, product development, strategy, negotiation, financing, deal structuring, international business, marketing, brand development and management, business development, public relations, press strategy, payments infrastructure, financial services, mergers and acquisitions, startups, software development, operations centers, board management, and investing.

Experience:

["Greylock · Full-time", "Inflection AI · Part-time", "Microsoft · Part-time", "Aurora · Part-time", "Joby Aviation · Part-time", "Coda", "NAUTO", "Entrepreneur First", "Village Global · Part-time", "Blockstream", "Reinvent Capital · Part-time", "Wolfson College, Oxford University", "Convoy Inc", "Neeva · Full-time", "OpenAI · Part-time", "Xapo"]

Education:

["Università degli Studi di Perugia", "University of Oulu", "The University of Oulu is an international science university which creates new knowledge, well-being and innovations for the future through research and education. The University of Oulu, founded in 1958, is one of the biggest and most multidisciplinary universities in Finland.", "Babson College", "University of Oxford", "Activities and societies: Wolfson College, Matthew Arnold Prize (Proxime Accessit)", "Stanford University", "Activities and societies: Marshall Scholar, Dinkelspiel Award, Golden Grant, Founder of the Symbolic Systems Forum", "The Putney School", "Activities and societies: X-country skiing, soccer, rebuilding Nova Scotia house"]

Figure 53 Test case 3.3

Posts:	
content	date
The best way to fix the housing crisis is to increase the stock of affordable, decent housing. People who don't understand economics tend to think, 'we can do price fixing and mandate what the prices are.' But that's simply not how markets work.	Wed, 03 Jul 2024 16:00:51 GMT
"By the standards of the rest of the world, basically every city here [in the U.S.] is a new city. And many of them are pretty recent. I mean, you look at Columbia, Maryland, or Irvine, California. They are 60 years old. ...They looked like cow pastures 60 years ago, and today they are these bustling places. So there's actual examples. It's not rocket science. ...We are not trying to invent a reusable rocket that lands. We are not trying to invent AI. It's become non-controversial to say that we can invent software that's as smart as humans or smarter, but somehow a lot of people today are saying, "we can't build some houses and factories and a sewer line." I mean, we've known how to do this for 5,000 years." - East Solano Plan / California Forever founder/CEO Jan Sramek For more from Jan, Aria Finger, and my conversation about the future of cities, including why I invested in California Forever:	Sun, 30 Jun 2024 20:00:03 GMT
One reason I'm an investor in California Forever—a project that's out to build a new, walkable city in Solano County—is because I believe that a big part of building the best possible future is thinking about the way people are living, connected, and finding community.	Wed, 26 Jun 2024 13:52:05 GMT
Part of the reason we've had economic prosperity in America is because we're fundamentally about the rule of law. Rule of law is best for business.	Tue, 25 Jun 2024 21:32:17 GMT
AI already is, and will continue to be, incorporated in restaurants and hospitality. But even as we see an increase in what the late John Naisbitt called "high-tech" components, many of the "high-touch" components that make us human aren't going anywhere. The experience of breaking bread together—the most human, important element—will always be front and center.	Mon, 24 Jun 2024 01:54:17 GMT

Figure 54 Test case 3.4

4) A user tries to search when his token is expired:

Enter LinkedIn Username

melrobbins

Include: ☒Name ☒Surname

Filter by Keyword:

No. of results:

5

localhost:8081 says

An error occurred during the search!

OK

Figure 55 Test case 4.1

```
false invalidId(
  bson.errors.InvalidId: 'Signature expired. Please log in again.' is not a valid ObjectId, it must be a 12-byte input or a 24-character hex
  string
}
```

Figure 56 Test case 4.2

References

- [1] "Social Media Data Service | Octoparse," [service.octoparse.com](https://service.octoparse.com/socialmedia).
<https://service.octoparse.com/socialmedia>
- [2] J. T. Richelson, *The U.S. Intelligence Community*. Routledge, 2018.
- [3] SuperSimpleDev, "HTML & CSS Full Course - Beginner to Pro," [www.youtube.com](https://www.youtube.com/watch?v=G3e-cpL7ofc),
Feb. 05, 2022. <https://www.youtube.com/watch?v=G3e-cpL7ofc>
- [4] Stefan Hyltoft, "Web Scraping in Nodejs & JavaScript."
<https://www.udemy.com/course/web-scraping-in-nodejs/>
- [5] Envato Tuts+, "Vue.js Tutorial: Beginner to Front-End Developer,"
[www.youtube.com](https://www.youtube.com/watch?v=1GNsWa_EZdw), Jan. 25, 2023.
https://www.youtube.com/watch?v=1GNsWa_EZdw
- [6] "The Selenium Browser Automation Project," *Selenium*.
<https://www.selenium.dev/documentation/>
- [7] Pandas, "pandas documentation — pandas 1.0.1 documentation,"
[pandas.pydata.org](https://pandas.pydata.org/docs/). <https://pandas.pydata.org/docs/>