



DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

BACHELOROPPGAVE

Studieprogram/spesialisering:	Vårsemesteret 2022
Bachelor i ingeniørfag / Datateknologi	Åpen
Forfatter(e): Genette Våge og Mina Wøien	
Fagansvarlig: Erlend Tøssebro	
Veileder(e): Terje Strømstad	
Tittel på bacheloroppgaven: Resource management tool	
Studiepoeng: 20	
Emneord: Online, Allocation tool, MVP, security, back-end, front-end , user friendliness, programming	Sidetall: 54 + vedlegg/annet: Zip-fil Stavanger 15. mai 2022

Contents

Content	i
Abstract	vi
1 Introduction	1
1.1 Task background	1
1.2 Implementation goal	1
1.3 Motivation	2
1.4 Workflow	3
2 Background	5
2.1 “Previous work”	5
2.2 Software technology background	5
2.3 Application Programming Interface	7
2.3.1 CV Partner	7

CONTENTS

2.3.2	Resource Guru	7
2.4	Azure Active Directory	8
3	Construction	9
3.1	Data models	9
3.1.1	Flowchart	10
3.1.2	Stack diagram	11
3.1.3	Site diagram	12
3.2	Code structure	14
3.2.1	Database structure	15
3.2.2	CV Partner API	16
3.2.3	Resource Guru API	17
3.2.4	Updating consultants information automatically . . .	18
3.2.5	Azure Active Directory	19
3.2.6	Uploading an engagement	19
3.2.7	Match engagements with consultants	20
3.3	Data Security	22
3.3.1	Storing secrets in Azure	22
3.3.2	Azure Active Directory	22
3.3.3	Validation of user input	23

CONTENTS

3.3.4	Personal data	24
3.4	User-friendliness	25
3.4.1	Buttons	25
3.4.2	Search	26
3.4.3	Sort instead of filtering	26
3.4.4	Routing to match	27
3.4.5	Skills view	28
3.5	Evaluation consulting	28
4	Results	30
4.1	Frontpage	30
4.2	Consultants	33
4.2.1	Consultant overview	33
4.2.2	Consultant information	33
4.2.3	Add new consultant	37
4.3	Engagement	38
4.3.1	Engagement overview	38
4.3.2	Engagement information	39
4.3.3	Add new engagement	42
4.4	Match	45

CONTENTS

5 Discussion	47
5.1 Environmental accounts	47
5.2 API	48
5.3 Performance of the application	48
5.3.1 API calls	48
5.3.2 Match function	49
5.3.3 Database indexing	50
5.4 Further Work	50
5.4.1 Evaluating consultants	50
5.4.2 Resource Guru API	52
5.4.3 Unassign consultants from engagements	52
5.4.4 Data security	53
Bibliography	55

Abstract

This thesis has been solved according to the need Cegal has. Cegal wanted us to create a product as part of a Cegal internal improvement project. The program should be able to assign resources to projects or engagements based on the consultant's availability and skills. In addition to these main categories, Cegal asked for a possibility to get the name of the engagement manager and a resource owner tied to engagement and resource. Cegal's last wish was to have the opportunity to add un-named consultants to the program for employees who have not started yet. The security requirements set by Cegal were that it should have strict access management, availability to managers only and General Data Protection Regulation (GDPR) compliance.

The main focus has been to streamline Cegal's allocation processes by developing an application that compares consultants' skills to skills needed in a specific engagement. In addition to comparing skills, showing the consultant's utilization has been a high priority. This resource management tool will help Cegal give the consultants the best-suited engagement and ensure that all the consultants are given engagements.

The code was developed in systems within Cegal's infrastructure. To develop this tool, it has used external Application Programming Interfaces (APIs) to collect information from already existing programs in Cegal, together with creating a solid and well-functioning code. The user experience has also been essential to ensure that the resource management tool will be used in the future.

Because this resource management tool contains personal data, security has

CONTENTS

been in focus from the beginning to the end. Azure Active Directory (Azure AD) has been used to maintain Cegal's login systems and especially for protection. By using Azure AD, the application is protected as it includes two-factor authentication. In addition, only the needed information is saved in the database to secure the application even more.

Chapter 1

Introduction

1.1 Task background

Cegal's consultancy unit is growing and is expected to increase by 100-200% in the number of resources in the coming years. Key stakeholders in sales, business development, engagement management, and resource owners need a unified tool to manage assignments and get transparency on returning consultants. These are consultants returning from excused leave, contracted duties, holidays, and sick leaves. To streamline resource allocation, Cegal needs a way to search for consultants based on competency, location, and availability.

1.2 Implementation goal

To simplify Cegal's resource allocation, the goal is to be able to make a program that assigns resources to projects or engagements. The managers of Cegal would be able to add new engagements or choose an existing engagement where consultants are needed. The engagements will be listed, and the user will be able to use a sort function to find the engagement they are looking for. The program will have a feature that matches the requirements from the demand with the consultants' skills. Consultants can be

1.3 Motivation

allocated to several engagements. Because of this, the program will state the availability and range them according to the number of matched skills.

The user can get more information about a consultant if needed. This could be information like which engagements the consultant is allocated to and how much they are utilized. By using Application Programming Interface (API), the program will receive pieces of information from other applications that Cegal already uses daily.

Cegal needed a way to know about new employees and have the ability to allocate them to customers from the day they started. To safeguard privacy requirements, these are entered with masking of names. Before these new employees are included in Cegal's systems, they are temporarily stored as manually added consultants in the database. This is done to have them in new projects quickly.

1.3 Motivation

Our motivation for this thesis is to help Cegal reduce their time finding the correct consultant for the right engagement. The second thing that motivates us is to make sure consultants are continuously assigned to engagements and minimize the waiting time until they get assigned to a new engagement. By doing this, our program will also help Cegal financially. The more consultants who are out with clients, the more Cegal can invoice.

After our first meeting with Terje Strømstad, we felt this application was something that would likely be beneficial for the business. Cegal just acquired Sysco and Envision, and the number of employees has increased significantly. Due to this, it has not been easy for the resource leaders to find consultants and know about their skills. The leaders in Cegal noticed this challenge and have since been committed to providing insights into business needs for the product we developed. Their engagement has motivated us to make an application that will cover the company's challenges.

After speaking with the employees, it dawned on us that the employees can also use this program in the future. But for them, the use of this program would be a bit different. For example, the employees in the company could

1.4 Workflow

hopefully use the program to look up other employees to find out who to ask for help when they need someone with a specific type of skill.

1.4 Workflow

Cegal works according to an agile framework called Scrum. This framework helps teams to work better together. After getting an introduction to how this worked, it got implemented into the workflow of this project. Scrum is a workflow that divides more significant tasks into smaller tasks and presents the projects more like an open book for those who work on that specific project [7]. It's normal to start building up a backlog with the Scrum team. In this project, the Scrum team consists mainly of the students themselves and the supervisors from Cegal. A backlog is a list of tasks required to support a larger strategic plan [4]. Typical items in a backlog are user stories, changes to existing feature, and bug fixes.

For our project, it was mainly user stories that were used, and the backlog got finalized by the students with the help of the supervisors to prioritize the items. The items ranked highest on the list represent the team's most important or urgent items to complete [4]. Target Process was used to keep track of the backlog and user stories. Target Process is an application that shows what you are working on and how far you have reached in your project. It is a great tool to use for planning one week sprints. Sprints are a short, time-boxed period where the Scrum team works to complete a set amount of work. In Scrum, the sprints can last from 1 to 4 weeks, but it was decided to work in one week sprints for this project.

An essential feature with Scrum is that you should be able to keep up with the project and catch up if a person in your team is stuck and needs help. That leads us to the daily stand-up meetings. Every morning fifteen minutes were set aside to go through yesterday's tasks and the plan for today.

At the beginning of the project, Monday, Tuesday, and Friday were set aside to work on the project. Mainly work together in person as we both operate better that way. It was decided to use LaTeX as the primary application for writing the report.

1.4 Workflow

When making an application for a company, the company must be able to develop the application further. Therefore, the GitHub placed in Cegals infrastructure was used to develop our code. By using GitHub, it's easier for a developer to develop the application further.

Chapter 2

Background

2.1 “Previous work”

In a previous bachelor thesis, a group of students has created a searching tool where a engagement demand is scanned and matched with the consultant’s skills from Sysco’s CV program. Due to compatibility and completeness issues, and different choices made for the software architecture, we chose to build on some selected functions written by the former bachelor student group. The inclusion of these accelerated our job, and freed time to solve more tasks in the backlog.

2.2 Software technology background

The chosen stack for the program is a PostgreSQL, Python, Vue and Node stack, as shown in figure 2.1.

2.2 Software technology background

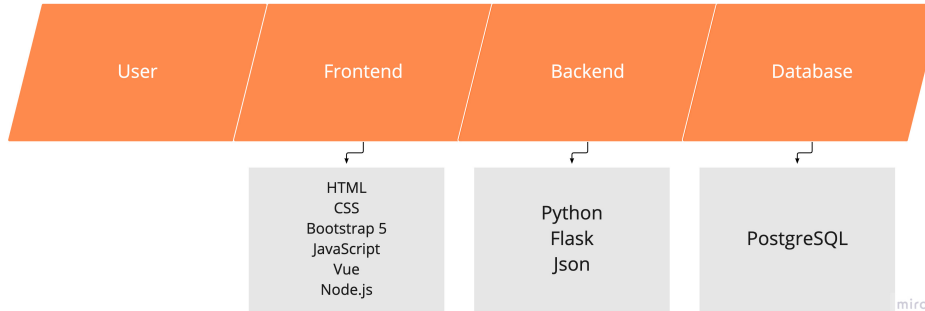


Figure 2.1: Development stack.

The thesis written by former bachelor students for Sysco, has used Python and Python Flask for the backend. The programming language Python was chosen because it contains good natural language processing libraries and is quick in data analysis [8]. Because the resource management tool is built on this code, it is natural to use the same programming language. Data scientists in Cegal use mostly Python for programming, and for frontend, developers use both React and Vue. In previous projects at the university, Vue has been used frequently for frontend programming. Therefore, it is a natural choice to use Vue in this project. Node is only used in the development stage to build the frontend for deployment.

Relevant information about consultants is retrieved with API calls to CV Partner and Resource Guru. The plan was to send API calls every time the information was needed. However, because these calls take time, the program required a database to store both the consultants and the engagements. The database chosen is PostgreSQL because it is open-source and contains many advanced features [19]. In addition, it is an accessible database that is easy to use when writing queries.

One decision that had to be made early is how much of the code from the previous project to use in the current project. As a result, only the most necessary algorithms were used instead of working on the previous student's code. The old code was difficult to work with, and it used deprecated Python packages. It would be very time-consuming to keep their code. However, a keyword handler class, that matches text input to a list of keywords, was made use of.

2.3 Application Programming Interface

2.3 Application Programming Interface

In order to integrate data from relevant programs, it was necessary to choose the main API to receive data from to get started. Students from the previous bachelor thesis only received the consultants from Sysco's CV Partner. Due to Cegal's extensive use and integration of CV Partner, the best solution for this project was to use an API from CV Partner. However, the API from Resource Guru was needed to get the consultants' utilization for a new engagement. Both the product we developed, and all programs our integrations connects to, are secured behind Cegal's cloud portal which is implemented in accordance with Cegal's Information Security Management System (ISMS).

2.3.1 CV Partner

The consultants are sourced with an API from Cegal's CV Partner. It is an application where the employees in Cegal can add their CVs. Each employee is automatically added to CV Partner when they start, with name, email, phone number, and department. The employees can also add a previous project, education, skills, etc. CV Partner is used to choose consultants for new engagements by looking at their skills, experience, and knowledge. Since the resource management tool matches the consultants' skills from CV Partner with engagements, the employees in Cegal must update their CVs regularly. Therefore, consultants should be motivated by the resource management tool to update their CVs as the program is dependent on it to work best.

2.3.2 Resource Guru

Previously when Cegal received an engagement demand for an engagement, they used an excel sheet to control where consultants were allocated. All the information about an engagement was written manually, and it was easy to make mistakes. The resource owner wrote the consultants' skills from CV Partner in the excel sheet to match them with engagements manually.

2.4 Azure Active Directory

In matching processes today, the resource owner uses a combination of Resource Guru and CV partner. The matching process itself uses CV Partner to look for which skills different consultants have that match the engagement description. When a match is found, they need to check the consultant's availability in Resource Guru. The resource management tool is more user-friendly and secure. It will make their allocation processes easier and less time-consuming.

2.4 Azure Active Directory

Azure AD is a cloud-based identity and access management service [1]. Cegal uses Azure AD services to ensure that their employees can access all external resources, such as Microsoft 365 and many other Software as a Service applications. It also helps the employees to access internal resources such as Cegal's cloud portal.

Management of Azure's services is restricted to IT admins. They will use Azure AD to control the employees' access to applications and applications resources based on business requirements. Some employees are not supposed to have access to specific programs, and an IT admin can control employee access using the Azure AD portal. Cegal uses two-factor authentication when accessing critical organizational resources.

App developers in Cegal use Azure AD as a standards-based approach for adding single sign-on to the applications they develop or use, allowing it to work with a user's pre-existing credentials.

The resource management tool program will be connected to Azure AD to make it easier for Cegal.

Chapter 3

Construction

This chapter considers the application's structure, decisions made during our work, and changes made.

It was decided to display and explain some data models to begin the construction chapter. In the presentation of our code structure, both the database structure and the different types of API will be explained. In addition to this, the code structure, the security developed because of personal data, the functions behind the file uploading and the matching process, will be described. By the end of the chapter, the decisions regarding user-friendliness and feedback from our consulting meetings, will be described.

3.1 Data models

Data models are visual representations of data elements and the connections between them. Models support the development of effective information systems by helping to define and structure data [5]. The data models explained in this chapter are flow chart, site charts, and stack diagram.

3.1 Data models

3.1.1 Flowchart

With a flowchart, you will view the whole process of the application quickly by using a few words and some figures. A flowchart will clearly show you what happens at each stage and how this affects other decisions and actions [2].

In the programs flow chart, as shown in figure 3.1, the application pages are shown as gray rectangles, and the buttons on the pages are explained as white rectangular boxes. Both form pages look like parallelograms. The parallelograms indicate the pages where you will have some input and output.

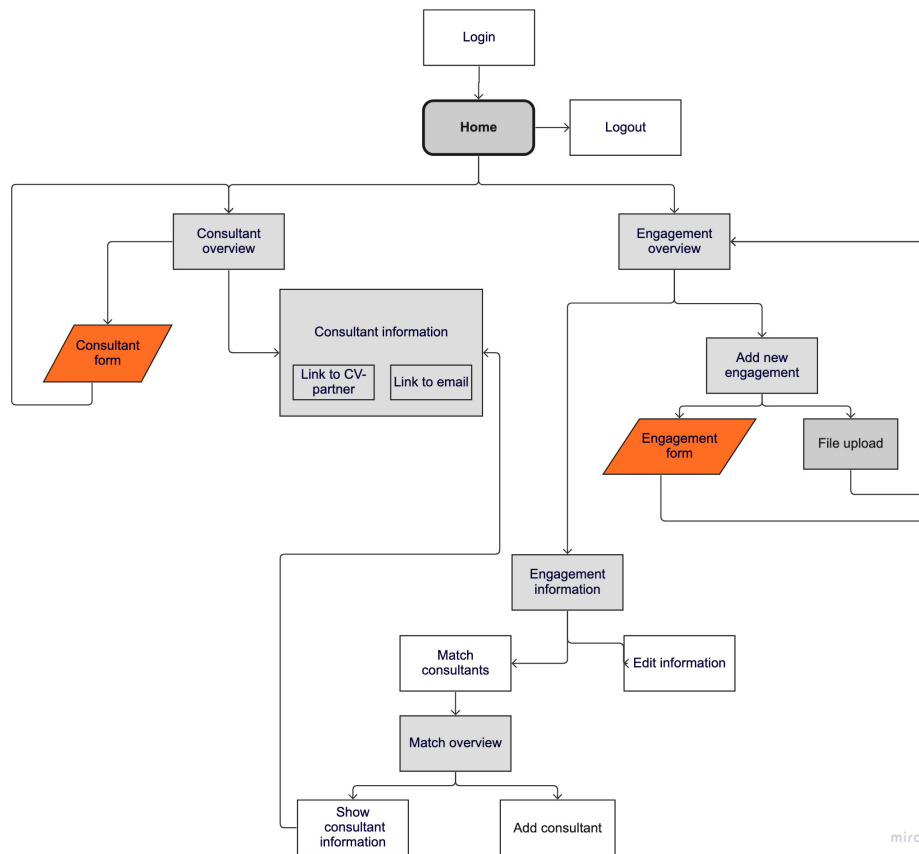


Figure 3.1: Flow Chart.

3.1 Data models

3.1.2 Stack diagram

In figure 3.2 we have visualized a frontend client that is communicating with the backend through API calls. The backend fetches data from external APIs, CV Partner and Resource Guru, and adds the needed data to the database before serving it to the frontend. The diagram visualizes how data moves around in the stack. It also pictures the responsibility of each part of the stack.

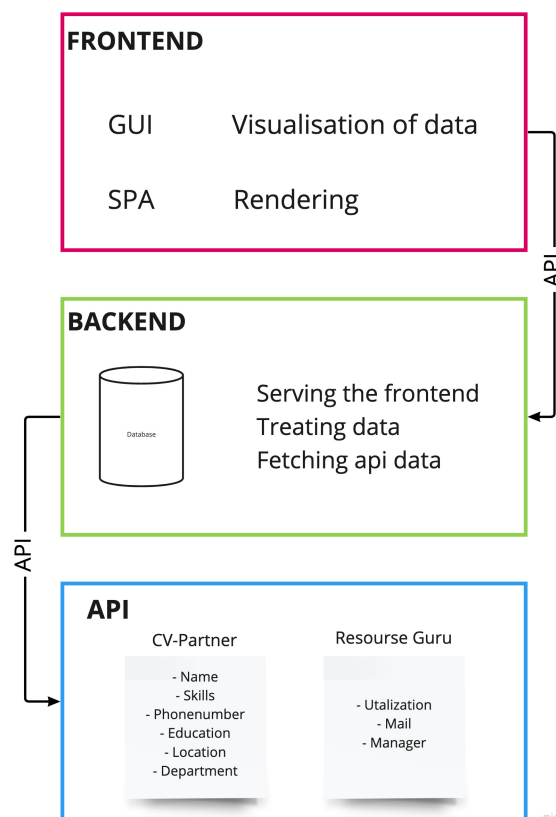


Figure 3.2: Stack Diagram.

3.1 Data models

3.1.3 Site diagram

The application will be a user-friendly application. The first page is shown to the left in figure 3.3. Here, the user has to log in to access the engagements and consultants. After the user has logged in, a welcome page is shown, as shown to the right in figure 3.3.

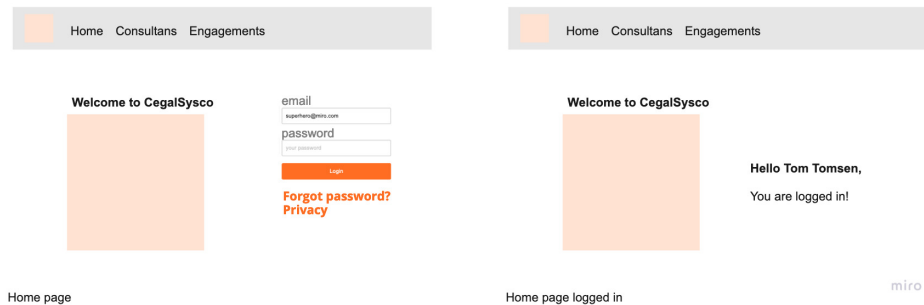


Figure 3.3: Site Chart for the Home page.

After the user has logged in, the user will access the engagement overview page, as shown to the left in figure 3.4. The overview page will display all the engagements the company has added. The plan for this page is to make it easy and clear to read, to help the user navigate quickly to the engagement it is looking for. From the overview page, the user will also be able to navigate to the “Add new engagement” page, the “match” page, and the “information about the engagement” page.

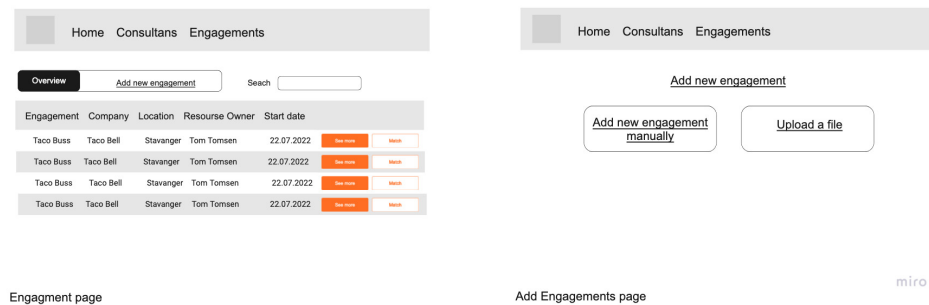


Figure 3.4: Site Chart for Engagements overview and Add engagement.

3.1 Data models

On the “Add new engagement” page, as shown to the right in figure 3.4, the user will be able to upload a PDF file that automatically retrieves the information about an engagement that is needed, or manually add the engagement information. The engagement form to manually add engagements are shown to the left in figure 3.5.

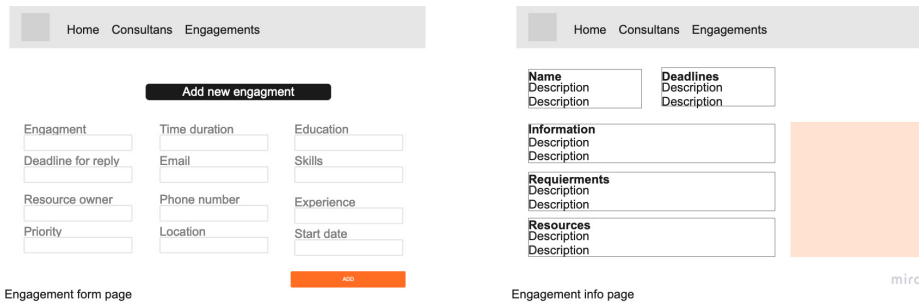


Figure 3.5: Site Chart for Engagement form and engagement information.

The “information about the engagement” page will be a simple page that contains all the information about the engagement, as shown to the right in figure 3.5. The idea is to make a function so that it is possible to edit the engagement information.

The matching itself will take place on the “match” page. On this page, you will get a list of all the consultants with matching skills as the engagement requires, as shown to the left in figure 3.6. The idea is that we add a type of filtering function to help the user navigate its way to the suited consultant.

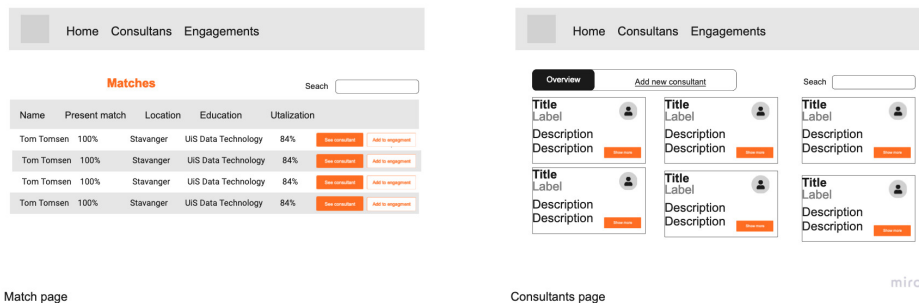


Figure 3.6: Site Chart for match page and Consultants overview.

3.2 Code structure

The consultant page will be roughly similar to the engagement page. In addition, the overview page will contain an overview of all the consultants, as shown to the right in figure 3.6, and there will be a search function to help the user navigate. This page will also guide the user to the “Add new consultant” page and the “information about the consultant” page, as shown in figure 3.7.

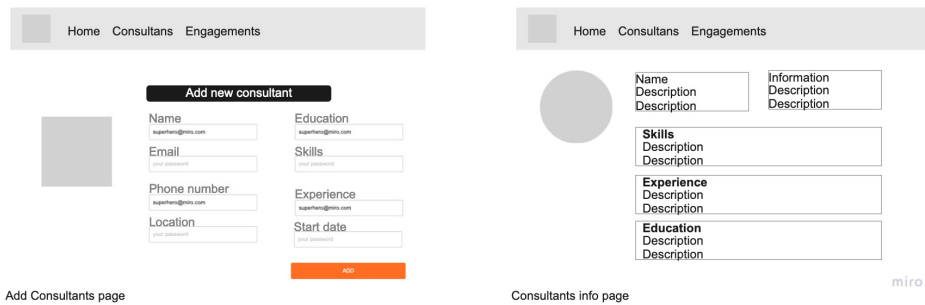


Figure 3.7: Site Chart for Consultant form and Consultants information.

Usually, all the consultants will be imported from CV Partner, but Cegal wanted a way to add and match so-called no-names consultants. No-name consultants are consultants that might not have started yet, which means they are not in Cegal’s systems yet. On the “Add new consultant” page, the user can add new consultants, as shown to the left in figure 3.7.

On the “information about the consultant” page, the user will get a simplified version of the consultants’ CV, as shown to the right in figure 3.7.

3.2 Code structure

The backend is written in Python flask, which connects to a PostgreSQL database, and serves a Vue frontend. To archive the information needed about each consultant, the backend sends API calls to both CV Partner and Resource Guru and stores the data in the database.

3.2 Code structure

3.2.1 Database structure

The database contains three tables, one table for consultants, one for engagements, and a table that maps between consultants and engagements. The last-mentioned table is required because consultants may have many engagements, and an engagement may have several consultants. Therefore, the many-to-many relationship between the two tables is split into the “ConToEng” table, as shown in figure 3.8. Such that data can be stored and retrieved without creating duplicates.

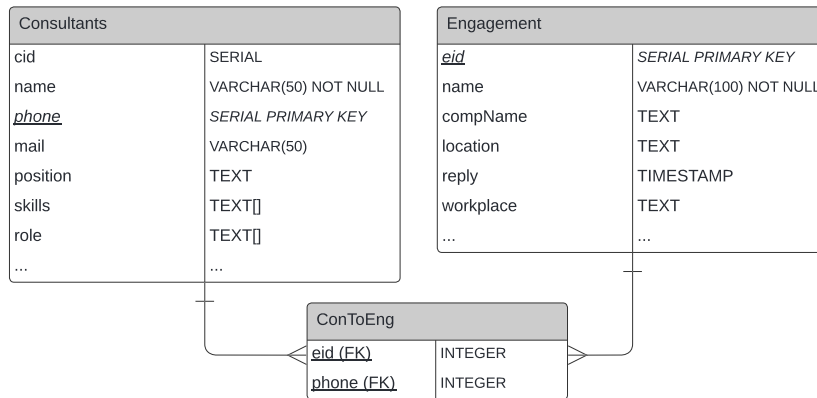


Figure 3.8: Database structure.

Cegal users can see a list of all the assigned consultants on the engagement information page. This list displays the consultant’s full name and phone number. This is done by using an advanced Postgres feature that aggregates table joins into a JSON object that can be directly sent to the frontend. The same method is used on the consultant profile site. As you can see in the figure 3.9, all the engagements a consultant is assigned to are listed for the user to see. A consultant is retrieved from the “Consultants” table in the database. By joining the “ConToEng” and the “Engagement” table with the “Consultants” table, we can build a JSON object with the information needed from the “Engagement” table. The engagements will be

3.2 Code structure

a list containing a JSON object for each engagement.

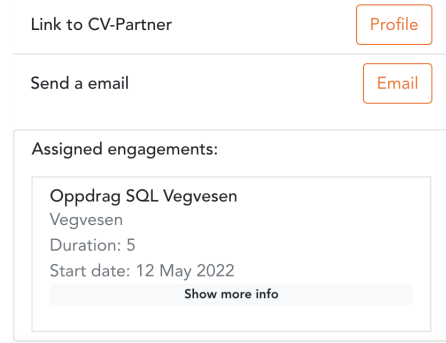


Figure 3.9: Consultant added to an engagement (Dummy data).

The best choice for the primary key in the “Consultant” table is phone number because consultants can update their information at CV Partner, and new consultants can be added. The user can manually add a consultant to make it possible to match a consultant that has signed a contract but has not started yet. This is to get the consultants an engagement quickly after they have begun. Manually added no-name resources will eventually have their first working day. Then they get registered in Cegal’s systems, including CV Partner, the consultant will be recognized by its phone number and updated in the database with the information from CV Partner.

Since the phone number and mail addresses are unique, they could be used as the primary key. However, because the manually added consultants do not have a Cegal mail address, it was more relevant to use their phone number. In addition, the risk of typos when entering data is higher for long strings. Therefore, it is also better to use phone numbers as the primary key. Another benefit of using phone numbers instead of mail addresses is that index structures for integers are more compact and faster.

3.2.2 CV Partner API

When data is retrieved from CV Partner, the backend will call the CV Partner API and update the database. The Python package retrieves all

3.2 Code structure

the users and an ID to get their CVs. Therefore, the application sends one API call to get all the users, and then another one for each user to get their CVs. Because their CVs includes a lot of information, only the information necessary for the application is stored in the database. It keeps, for example, name, phone number, mail, workplace, skills, education, and former projects.

When the frontend or backend needs data from the API, the backend serves data from the database instead of calling the API. This is done to speed up data retrieval and reduce the number of API calls.

3.2.3 Resource Guru API

Knowing whether consultants are available or not for new engagements on a given date is mandatory for this application. The utilization for each consultant is retrieved from Resource Guru.

When the consultants' resource IDs are retrieved from Resource Guru, an API call is first sent to Resource Guru to find all the resources. Afterwards, the API call responds with a small summary from all the consultants, containing the resource's name and ID. Finally, after retrieving the ID, the ID is used in an API call to fetch the consultant's email. This has to be done for each resource.

Many consultants have not registered their phone number at Resource Guru. Therefore, the consultant's utilization is added to the database by recognizing the consultant's mail address.

For each consultant that matches an engagement, an API call is sent. The query parameters in the API calls are the start- and end date of the engagement and the resource ID for each consultant. Finally, the request returns a report for each consultant containing their utilization between the start- and end date.

The manually added consultants are registered with 0% utilization from their start date. They will not be updated with availability from Resource Guru before they have received a Cegal mail and are added to all of their systems.

3.2 Code structure

3.2.4 Updating consultants information automatically

Getting information with API calls from CV Partner and Resource Guru is time-consuming. The data has to be updated regularly as the consultants can edit their information at CV Partner, and new consultants can be added to Cegal's systems. Instead of having an update button as first planned, a function that updates the database by sending API calls once a day where created. With the use of the multiprocessing package in Python, the package lets us use multiple processes simultaneously [15]. It is used to run the update function for the API calls periodically. The API to CV Partner that fetches the user will be called every 24th hour. Then a function checks whether the user's phone number is already added to the database. The consultant will be updated if the phone number is in the database. However, if a manager or a resource owner adds a new consultant to CV Partner, the consultant will be added if the phone number is not in the database. As the consultants have their phone number as the primary key in the database, no consultant will be added twice. The manually added no-names consultants will have their phone numbers in the database. Therefore, they will be updated the same way as the other consultants retrieved from CV Partner.

Not all consultants have written their phone number with their country code in CV Partner. As Cegal have offices in other countries, the country code is necessary when the phone number is called. Since the phone number is stored in the consultant table as an integer, the country code is saved independently in the table. Phone numbers have to have a minimum length of eight. However, it can be longer than eight integers as the code will check whether the user added the country code or not. The phone number gets stripped for spaces. When the length exceeds eight, it is split by leaving the last eight characters as the phone number and letting the remaining in front be the country code. However, if the length equals eight characters, the country code is found in a function. With Python's geopy package, their office location discovers the consultants' country. Then, the country is used to match with the countries in a country code file, containing countries and their country code. Finally, the phone number is converted to an integer before being added to the database.

3.2 Code structure

3.2.5 Azure Active Directory

Before we could enable Azure AD in the resource management tool, we needed to create a project in the portal of Azure AD. The site gave a client ID, a client secret, and a code example for Python flask. However, as the resource management tool uses Vue in the frontend, this code had to be changed to support it. The client ID, client secret, and a redirect URI were altered to be the same in the code as at the portal.

By redirecting to the login for Cegal, the managers can log in with their Cegal account, and their user gets stored in a cookie session until they log out. Then, when the site is loaded, a function will check if there is a user in the session, and if not, the person will have to log in to access the other sites.

When a user uploads an engagement or adds a consultant to the program, the logged-in user will be set as manager for the consultant and the responsible person for the engagement, as shown in figure 3.10. The user is retrieved from the session and added to the database.

Responsible	
Name:	Mina Woien
Email:	Mina.Woien@cegal.com

Figure 3.10: Responsible for the engagement.

3.2.6 Uploading an engagement

To find the best suited consultants for requested engagements, the sales personnel can upload a PDF file and match it with the consultants' skills. To prevent Cross-site scripting [3], the extension check whether it is a PDF or not. Cross-site scripting means that a possible attacker would not be able to upload a malicious script [10]. Cross-site scripting can be avoided by checking if the filename ends with ".pdf" and with python's PDF2 PDF File Reader package [17]. It checks if the file is a readable PDF file and catches possible exceptions. If it is not legible, the user will get an error

3.2 Code structure

saying the file was not uploaded successfully.

The file is uploaded using FormData in Vue, which encrypts the file and sends it to the backend. Then, by writing bits into another file on the local server, the file can be stored as plain text in the database. Further, the function opens the local file and uses the PYPDF2 Python package to read the file. This package has a function that gets the number of pages, and by looping over the pages, it can extract the text from each page.

The text gets translated to English, with the Translator class from google-trans package, as the consultants' skills are written in English. Before the engagement is added to the database, a function finds the duration, start date, location, and name of the company's engagement. All this information is retrieved by the "find()" function in python. The function searches for common words that are placed in front of these variables. The function used to find duration looks for the words "Months", "Year", "Duration", "Duration long" and "Contract Period". Users can edit the engagements if this function does not find the necessary information and add more information about the engagement.

3.2.7 Match engagements with consultants

The application matches consultants' skills, skills from previous projects, and education degree with engagements. The match function checks whether the engagement has a description or not for the manually added engagement, as it is only required to add required skills for the engagement. If the engagement has a description, the function will first find matching keywords and then add the keywords that match the required skills.

Some engagements do not have education requirements, but for those engagements that require a specific education degree, for example a bachelor or master's degree, the matching page will list whether the consultant fulfills this requirement or not, as shown in the red square in figure 3.11. Consultants education is retrieved from CV Partner, and if the engagements has an education degree required, a function checks if the consultants' have this type of education degree. It will also say that the degree is satisfied if the engagement requires a bachelor's degree and the consultant have a master's degree.

3.2 Code structure

Consultant	Skills ↑	Utilization	Location	Education requirement	Position
Lars Hansen	3 Show skills	0%	Oslo	Not satisfied	Show more info Add consultant
Mona Monsen	azure sql Hide skills	0%	Stavanger	Satisfied	Show more info Consultant added
Henrik Hansen	2 Show skills	0%	Lysaker	Not satisfied	Show more info Add consultant
Katrine Hille	1 Show skills	0%	Stavanger	Satisfied	Show more info Add consultant

Figure 3.11: Consultants matched with an engagement (Dummy information).

The skills are displayed in numbers for user-friendliness as the consultants can match many skills with the code from the previous bachelor thesis. By clicking on the button “Show skills” as shown in the blue square in figure 3.11, all the skills are displayed. These skills are both skills the consultants have written at CV Partner and from previous projects. Because skills can appear duplicated with the function from the previous bachelor thesis, a function to remove the duplicates were created to save the skills in a list without duplicates. Duplicates can happen if the consultants have added skills in both CV Partner fields. Without removing the duplicates, the number of skills on this page would be misleading.

Before an engagement can be matched with consultants, it needs a start date and a duration. The Resource Guru API requires a start date and an end date for the engagement to get the consultants’ utilization. The start date for uploaded engagement can be found when the user uploads a PDF or when the user manually adds the start date. If the engagement does not have a start date or duration, an error message will be displayed when a user tries to match the engagement. The end date is found in a function using Python’s Dateutil Relativedelta package, using the start date and the duration of the engagement. The function is called when a user edits the engagements or when it matches the engagement with consultants. Then it can use it to get the utilization from Resource Guru when the user matches an engagement. For manually added engagements, the start date and duration are required fields.

3.3 Data Security

Before matches are displayed, the match function checks whether the consultants have started working at the company or not when the engagement is about to begin. Then the matching page will only list the consultants that have started working when the engagement begins. The manually added consultants are displayed with a 0% utilization when they do not have any engagements assigned.

3.3 Data Security

Since the resource management tool processes personal data about the consultants in the company, the site has to be secure.

3.3.1 Storing secrets in Azure

Keys with identifying information are needed to be able to make APIs calls [13]. For example, the CV Partner API requires only an API key. However, the Resource Guru API needs both an API key and a secret key. The API key is a public identifier for the application, and the secret key is used to authenticate and make requests to the application. Both of these keys are used to confirm the application's identity. The secret key must not get exposed, and storing secrets is risky [11]. These keys need to be stored and used securely, and therefore, they are stored in the Azure AD Portal. Azure Key Vault offers safe storage of secrets, and the application can easily and securely access the keys without storing them in the source control [11].

3.3.2 Azure Active Directory

For authentication, Azure AD is used to identify the users. Since not all employees at Cegal will have access to the resource management tool, Azure AD only allows a restricted group to access the program. This restriction is created by making a group in the Azure AD portal. This group only contains the few users that have permission to use this program.

It is impossible to create a user at the website, as the user will use their

3.3 Data Security

Cegal account. Since Azure AD takes care of the login, the user password does not have to be stored in the application. If we did not use Azure, we would have to hash the password before storing it in the database to resist possible attackers from accessing it.

Two-factor authentication prevents a possible attacker from getting the same privileges as the users. Two-factor authentication is used in case an attacker, for example, gets a hold of the passwords of one of the users [21]. The implementation of Azure AD includes a two-factor authentication with the user's password and the user's phone. The two-factor authentication helps to prevent an unauthorized person get access to the user accounts [14].

3.3.3 Validation of user input

To prevent cross-site-scripting by a possible attacker, the user input gets validated. By implementing Flask's "escape()" function in all input fields at the site, the special characters are converted into HTML-safe sequences [21]. However, since the escape function does not catch all the special characters, the remaining ones are placed in a list, as shown in the code 3.1.

Kode 3.1: forms.py

```
s avoid = ['$', '%', '=', '{', '}', '[', ']', '\\', '*', ...
         '^', 'Ã', '¬', '']
```

Each input field in the application, that is not an integer, loops through this list. If the input contains one of these characters an error message will be shown, as shown in figure 3.12, and the user will have to fix the error before sending the form.

Add new consultant

Name does not allow special characters, try again!

Figure 3.12: Error message from validation of user input.

3.3 Data Security

Some of the input fields are required, as they are needed for the manually added consultants or the manually added engagements. For example, the start date and duration for engagements are fields that are required, as explained in chapter 3.2.7. These fields are marked with a red star on the form. If the user tries to submit the form before these fields are completed, an error message will, in your computer language, appear at the required input fields. All the input fields for integers do not allow numbers under zero. The number of resources is needed, so the field is assigned to not allow numbers under one, as shown in figure 3.13.

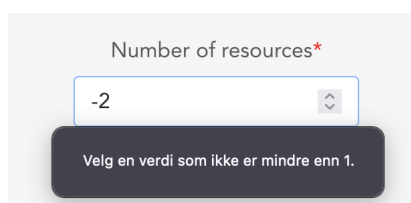


Figure 3.13: Error message when entering negative value.

3.3.4 Personal data

As the application stores personal data about the consultants in the database, GDPR principles must be followed.

According to GDPR, information that can refer to a person, for example, name, picture, email address, previous or current location, or health information, is considered personal information [18]. Therefore, as the program retrieves consultants from Cegal with CV Partner, information that can relate to one of the consultants is removed from this report.

The program has been developed according to GDPR's security of processing personal data. Using Azure AD and only letting a few selected people use the resource management tool makes sure that the program's ongoing confidentiality, integrity, availability, and resilience are secure [9].

Another way to improve security and comply with GDPR principles is by pseudonymization and encrypting personal data [9]. Unfortunately, this has not been done and is future work.

3.4 User-friendliness

Only the needed data is stored in the database, preventing possible attackers from getting data that is not required. For example, no passwords are stored in the database. The consultants in the company can also change their information at CV Partner, and the information will be updated in the database within 24 hours.

3.4 User-friendliness

This application is built on the fact that it will make it easier and more user-friendly for the engagement owners. However, as we have told previously, Cegal uses multiple applications such as CV Partner and Resource Guru. The goal of this application is to collect necessary information from the different applications and integrate that into our application.

This application will, in the beginning, only be used by people who work with sales and consultant allocations processes. That means that the application has to be built for these people and their needs. Even though this application has a scaled audience, there are some user experience (UX) guidelines that we are encouraged to use. Therefore, it was decided that a meeting with the UX responsible for Cegal was necessary to follow Cegal's policies. As a result, several decisions were taken to improve the application's user-friendliness.

3.4.1 Buttons

When you are working with user-friendliness, buttons are crucial. We did not have a plan for when the buttons should look a certain way in the application. The buttons were just styled and placed randomly, so to improve the user-friendliness, it was decided that different meanings of the buttons should have different styles.

Buttons that have a purpose of navigating you to other pages have a simple design. These buttons are there to route you to other pages in the applications but are not to draw your attention to them, as shown in figure 3.14. You will see them change color when you hover over them.

3.4 User-friendliness

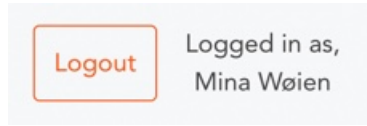


Figure 3.14: Simple button.

As you can see on button in figure 3.15, the action button is there to draw your attention towards them. These types of buttons are used on pages where the next step should be to push these buttons. For example, uploading an engagement manually or adding a new consultant.

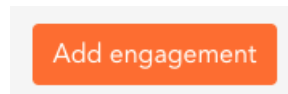


Figure 3.15: Action button.

3.4.2 Search

To help the user easily navigate through all the engagements and consultants, we added a search box on the three overview pages. One for consultants where you can search among the consultant's names, and the two other search boxes are placed on the "Engagement Overview" page and the "Match" page. On these pages, you can search through everything shown in the overview, anything from the start date to the name of the resource owner. Usually, the search box was placed in the main menu at the top of the page. We decided to remove this box and have smaller containers on each page to help the user understand the purpose of this search box.

3.4.3 Sort instead of filtering

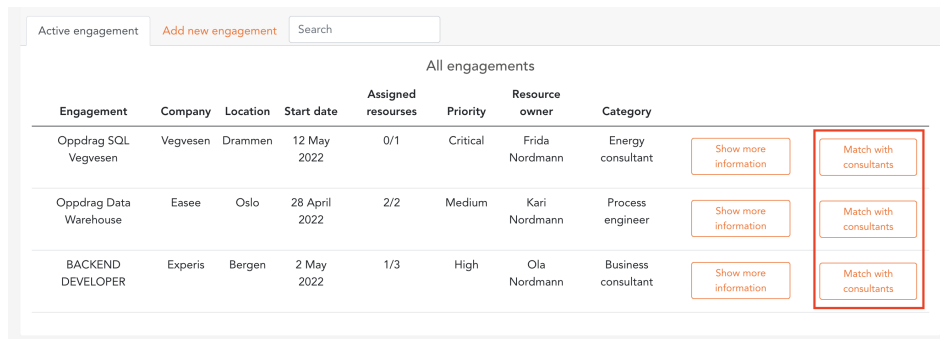
At the beginning of the project, the plan was to apply and add filtering features for the application. The goal was to make it possible to filter on skills, location, education, and experience on the matching page. However, after attending some resource management meetings with the leaders and resources owners, which is the people that will use our application, we clearly

3.4 User-friendliness

saw that our goal with filtering would not match their needs. Therefore it was decided to use a sorting function and add a search box on the matching page. The sorting function sorts the number of skills and percent utilization from highest to lowest. When entering the matching page, it is sorting from highest to lowest number of skills a consultant has by default. The user can also sort alphabetically on engagement name, location, education and position.

3.4.4 Routing to match

After testing for some time, we quickly saw the need to have a “match with consultant” button on the overview page, as shown in the red square in figure 3.16.



Engagement	Company	Location	Start date	Assigned resources	Priority	Resource owner	Category		
Oppdrag SQL Vegvesen	Vegvesen	Drammen	12 May 2022	0/1	Critical	Frida Nordmann	Energy consultant	Show more information	Match with consultants
Oppdrag Data Warehouse	Easee	Oslo	28 April 2022	2/2	Medium	Kari Nordmann	Process engineer	Show more information	Match with consultants
BACKEND DEVELOPER	Experis	Bergen	2 May 2022	1/3	High	Ola Nordmann	Business consultant	Show more information	Match with consultants

Figure 3.16: Match button from engagement overview page (Dummy data).

The “match with consultant” button was initially only placed on the engagement information page. However, instead of routing the user through two pages, adding an extra button on the overview page was decided. Adding this to the overview page will help the user understand the application better and how it works. An error will occur at this page if the engagement do not have a start date and/or a duration, as explained in chapter 3.2.7.

3.5 Evaluation consulting

3.4.5 Skills view

Some extra functionalities were added to the skills column for user-friendliness and the program's functionality. As explained in chapter 3.2.7, the skills are sorted from highest to lowest by default. We integrated a button that makes all the consultant's skills visible to spare the user time, as shown in figure 3.11. Our first layout showed all the skills a consultant matched with on the overview site. This worked when there were no more than five skills per consultant. As soon as this started to scale up and each consultant got ten to fifteen, the overview page became disorganized and lost its intention. To still give the user the option to view the consultants' skills without going to the detriment of making the page unclean, we integrated this "show skills" button.

3.5 Evaluation consulting

After observing a couple of Cegal's regular resource and engagement meetings, it is easy to see their need for a resource management tool. Today, assigning and finding the most suited consultants to incoming engagements is done mostly manually, as explained in chapter 2.3.2.

Several consultants have days where they aren't fully exploited. This is poor utilization of the consultants and leads to less income for the company. The consultants that are first discussed are usually the consultants that are already booked for other engagements. This problem demonstrates their need for a resource management tool.

One of the improvements this meeting led to was the thought of giving the engagement a resource owner, and a responsible person for the engagement. The resource owner is the person accountable for the human resources. The resource area where the resource owner primarily originates is where they have an overview of all the competence and personnel. Resource managers allocate according to development plans and strategic priorities. The person responsible in this case is the person who uploaded the engagement to the application. It was necessary to get information about who uploaded the engagement in case of questions regarding it. To optimize this process and

3.5 Evaluation consulting

save the user some time, we decided to automate this process. When a user is uploading or manually adding an engagement, their name and phone number will automatically be retrieved from the profile logged in through Azure AD, as explained in chapter 3.2.5. This contact information is placed under the “responsible” section on the engagement information page, as shown in figure 3.10. By doing it this way, missing contact information from the person who uploaded the engagement will never be a problem.

For the matching process, it emerged that the person in charge of allocating consultants to engagements would like to see the position of the consultant. Showing if the consultant is a senior consultant or a trainee helps the allocator decide what type of resources they will use for that specific engagement. For example, some engagements do not require senior technical input, which means a senior consultant with a high average rate is not optimal for that engagement. Instead, a trainee could get the opportunity to experience how it is to work in the field. Another example is if the rate for the engagements is low, it is not optimal to hire a senior consultant for this engagement.

The merger of Cegal, Envision and Sysco has led to a significant increase in consultants. Going from knowing all the consultants in the company to not having a complete overview of all the consultants and their skills puts an end to the old allocation process. It was important for the people that work in these processes to quickly get the information about who the manager of the consultant was.

Chapter 4

Results

This chapter will explain the resource management tool application and its use. To get a better overview and understanding, we will include some illustrations of each page. Then, for each page, we will explain what it is and the functions each particular page has.

4.1 Frontpage

The first page you will see when you enter the application is the home page, as shown in figure 4.1. This page is a clean page with no particular info other than that it asks you to log in. Because this application is only for people who work in Cegal, as explained in chapter 3.3.2, there is no place to register as a user. The only way to log in to this application is by using your existing Cegal email and password.

4.1 Frontpage

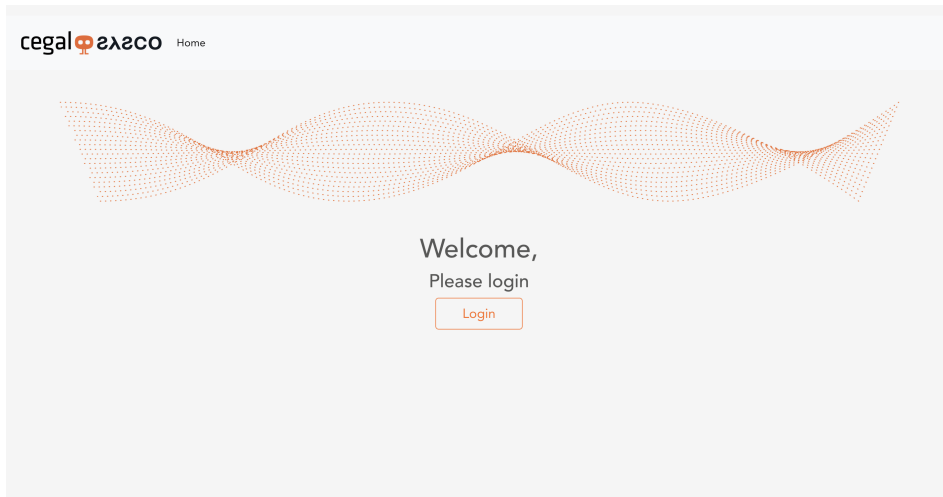


Figure 4.1: Home page when the user is not logged in

After the user has pushed the “login” button, they will be redirected to an Azure AD site, as seen in figure 4.2. If the user is registered in the group for the people who will have access to this application, they will be able to log in as if it was your everyday log-in routine. If they do not have access to this application, they will get a message saying they do not have permission to log in.



Figure 4.2: Azure AD Login

4.1 Frontpage

When the user has logged in, there will be more to see and buttons to push. The home page contains the same illustration as the first homepage has, but with some more information, as shown in figure 4.3. In a meeting with the UX responsible at Cegal, we were told to insert some more details on this application. The text we added was a short description of the application and what the user will mainly use it for.

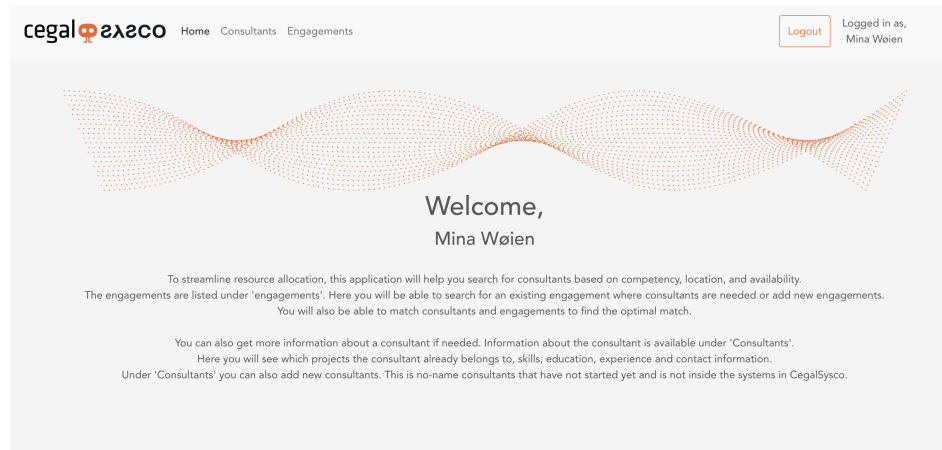


Figure 4.3: Home page when the user is logged in

The logged-in user will also get more options in the menu bar. For example, the consultants and engagements options will be viable next to “home”, as shown in figure 4.4. These buttons will guide you to either the consultant’s overview page or the engagements overview page.

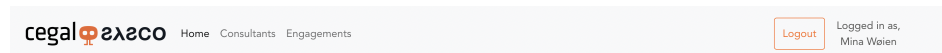


Figure 4.4: Menu bar

To the right in figure 4.4, the user gets the option to log out. In addition to the “logout” button, a short text is added to show the logged-in user’s name.

4.2 Consultants

4.2 Consultants

4.2.1 Consultant overview

The consultant's button in the menu bar will guide you to an overview page. This page contains a complete overview of all the consultants Cegal has, as shown in figure 4.5. The information shown on the overview page is a photo, either a picture of themselves or a standard user picture, the consultant's name, department and location.

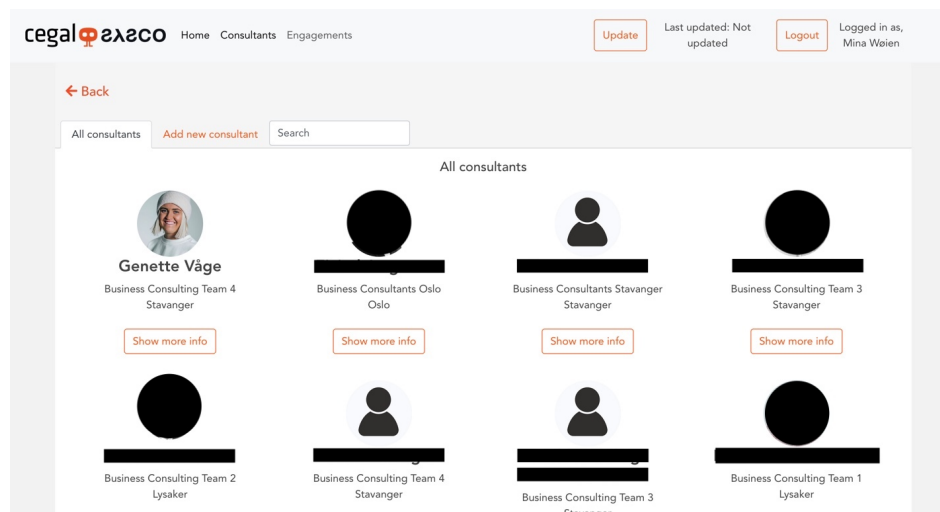


Figure 4.5: Consultant Overview page

A search box has been added to this overview page to make it easier to navigate, as explained in chapter 3.4.2. The user can use this search box to search for the consultant's name.

4.2.2 Consultant information

The "Show more info" button in figure 4.5 redirects you to an information page. This information page is a summary of the consultant's CV, as shown in figure 4.6.

4.2 Consultants

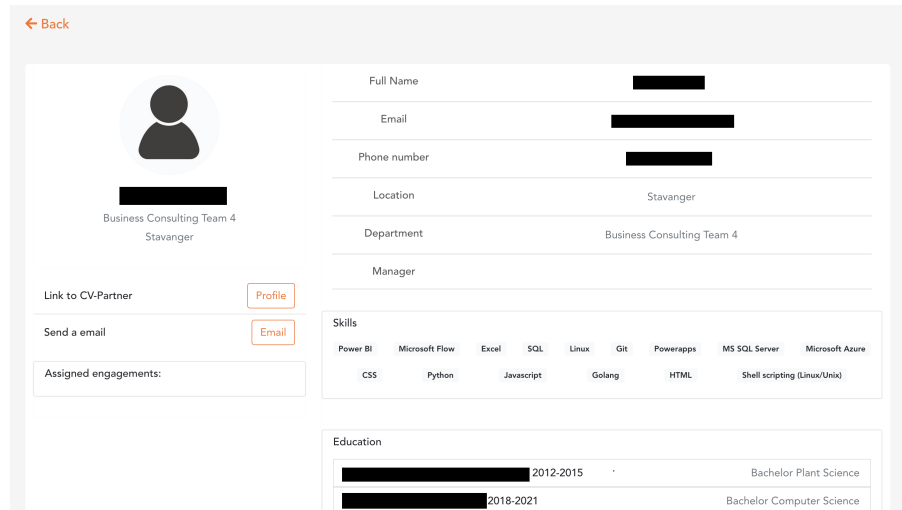



Figure 4.6: Consultant Information page

The information shown on this page is the most necessary information the user would need when allocating the consultant to an engagement. However, if the user wants to get more information, a button placed under the picture and the consultant's name will take you to the consultant's profile at CV-Partner, as shown in figure 4.7. In addition, a "Send email" button to make it quick and easy for the user to contact the consultant.

4.2 Consultants



Genette Våge
Business Consulting Team 4
Stavanger

Link to CV-Partner	Profile
Send a email	Email

Figure 4.7: Profile picture with links on the Consultant Information page.

Under the two buttons, in figure 4.7, a section called assigned engagements has been added. Whenever a consultant is assigned to an engagement, the engagement will be added to this section, as shown in figure 4.8. It will contain information like the engagement name, duration, start date, and company name. In addition to the consultant information, there is also an information page for engagements. To quickly navigate to the engagement information, a “Show more info” button to guide the user to this site has been added.

4.2 Consultants

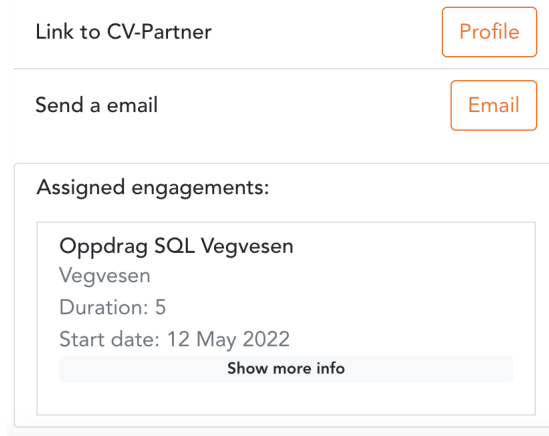


Figure 4.8: Consultant assigned to an engagement on the Consultant Information page (Dummy data).

In the top right area of the information page, some basic information like the name, phone number, location, department, and manager is listed, as shown in figure 4.9.

Full Name	Genette Våge
Email	genette@hotmail.com
Phone number	+47 12345678
Location	Oslo
Department	Business Consultant
Manager	Mina Wøien

Figure 4.9: Consultants information (Dummy data).

The following section on the information page is the skills, education, and experience, as shown in figure 4.10. The skills section will give the user a good overview of all the skills a consultant has added so that it is easy for the user to see if the consultant has the right skills. The different types of de-

4.2 Consultants

grees a consultant has are listed in the education section. It says what kind of degree, which year the consultant studied, and the name of the college or university. The last section on the information page shows the experience a consultant has. These experiences are typical of previous projects they have participated in, so it will show the name of the engagement and the name of the company.

The image shows three sections of a consultant's profile:

- Skills:** A collection of skill tags including Power BI, Microsoft Flow, Excel, SQL, Linux, Git, Powerapps, MS SQL Server, Microsoft Azure, CSS, Python, Javascript, Golang, HTML, and Shell scripting (Linux/Unix).
- Education:** A table with two rows:

Norwegian University of Life Sciences, 2012-2015	Bachelor Plant Science
University of Stavanger (UiS), 2018-2021	Bachelor Computer Science
- Experience:** A table with four rows:

IT Health Check	Cegal AS
EnergyCo	Cegal AS
Infosec Maturity Assessment	Cegal AS
CIP Application	[REDACTED]

Figure 4.10: Consultant's skills, education and experience on the Consultant Information page.

4.2.3 Add new consultant

The consultant overview page can also direct you to a page where the user can add a new consultant, as shown in figure 4.11. As explained in chapter 3.1.3, these consultants are so-called no-name consultants. No-name consultants are consultants that have signed a contract but have not yet started. The user can manually add the consultant to include these consultants in the matching process.

4.3 Engagement

Add a new consultant

Name*
enter name

Mobile Number*
enter phone number

Email
enter email

Location*
enter location

Department
enter department

Experience and skills

Experience
enter experience

Skills*
enter skills

Education
▼

Start date*
dd . mm . yyyy

Add new consultant

Figure 4.11: Add new consultant page.

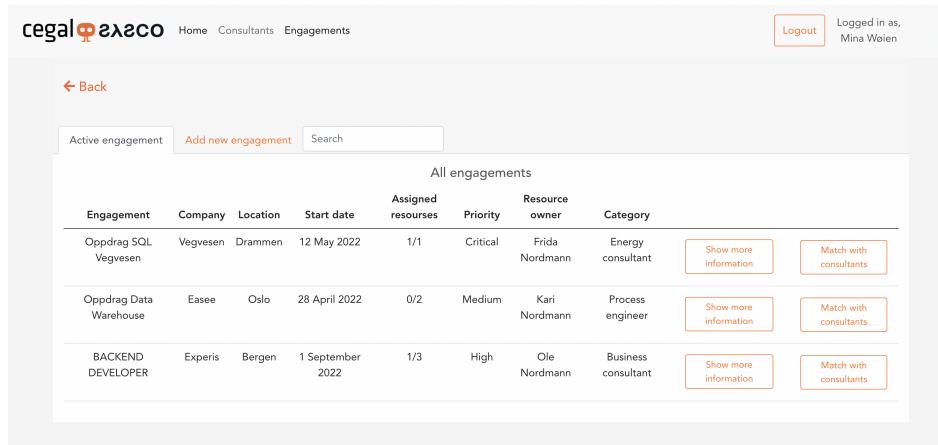
Some input fields are necessary for the user to fill in. These types of inputs are marked with some small red stars. When all the required fields are completed, the user pushes the “Add consultant” button, and the page will redirect the user to the consultant overview page. An error message will show if the required fields are not filled, as explained in chapter 3.3.3.

4.3 Engagement

4.3.1 Engagement overview

The engagement overview is almost identical to the consultant overview page. The only difference is that all the engagements are listed downwards, as shown in figure 4.12. It was essential to get some feedback from the potential users about the most necessary information to see. The information the potential users wanted to see was implemented into this overview table.

4.3 Engagement



Engagement	Company	Location	Start date	Assigned resources	Priority	Resource owner	Category		
Oppdrag SQL Vegvesen	Vegvesen	Drammen	12 May 2022	1/1	Critical	Frida Nordmann	Energy consultant	Show more information	Match with consultants
Oppdrag Data Warehouse	Easee	Oslo	28 April 2022	0/2	Medium	Kari Nordmann	Process engineer	Show more information	Match with consultants
BACKEND DEVELOPER	Experis	Bergen	1 September 2022	1/3	High	Ole Nordmann	Business consultant	Show more information	Match with consultants

Figure 4.12: The Engagement overview page (Dummy data).

Two buttons on the side for each engagement have also been added, as shown in figure 4.12. The first button redirects the user to an information page about the engagement. This page is in case the user needs more detailed information about a specific engagement or wants to edit the engagement information. The second button will redirect the user immediately to the matching process for a better user experience, as explained in chapter 3.4.4.

Equal to the consultant page, a search box has been added. This will help the user to navigate easier, as explained in chapter 3.4.2. In addition, some more search options to this search box has been added. For example, the user can use this search box to search for the engagement name, company, location and start date.

4.3.2 Engagement information

If the user wants to get more information about a specific engagement, they can press the button that guides them to the engagement information page. This page contains all the information an engagement should have or typically got, as shown in figure 4.12. The most necessary information regarding the engagement is placed in the first white section box at the top.

4.3 Engagement

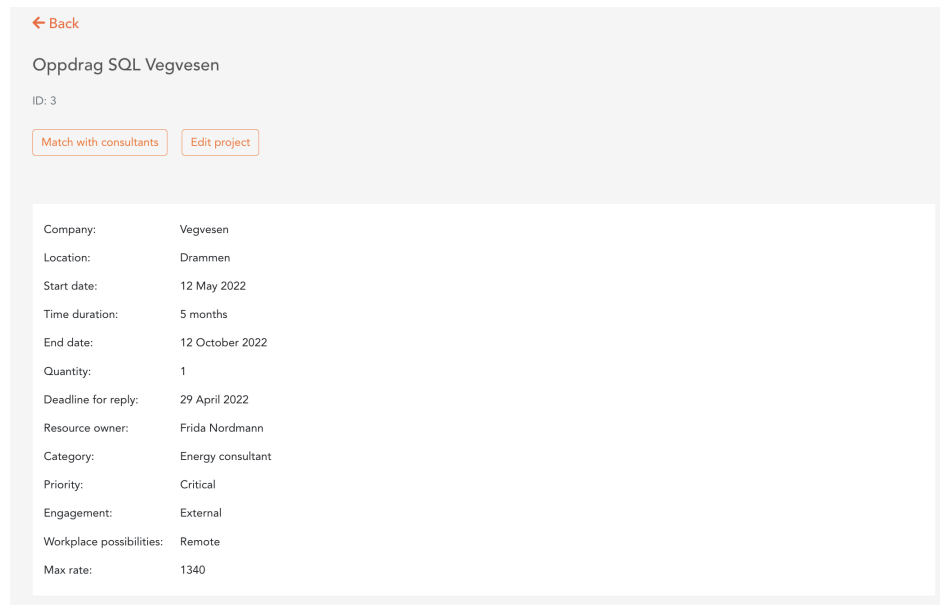


Figure 4.13: Engagement information page.

The following section box is for the requirements the engagement has, as shown in figure 4.14. This requirement is typically how much a consultant will be allocated, what education degree they require, and lastly, the skills the company mandate.

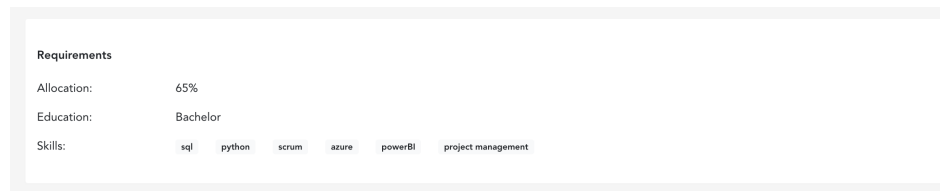


Figure 4.14: Requirements on the Engagement information page (Dummy data).

Since more than one user will be using this application, it was essential to have a section showing the responsible for a specific engagement. The person responsible for the engagements is the person who uploaded the engagement to the application. The user's name and email is automatically filled in when uploading an engagement, as shown in figure 4.15. A section like this has been added to help the user know who to contact regarding

4.3 Engagement

questions related to the engagement.

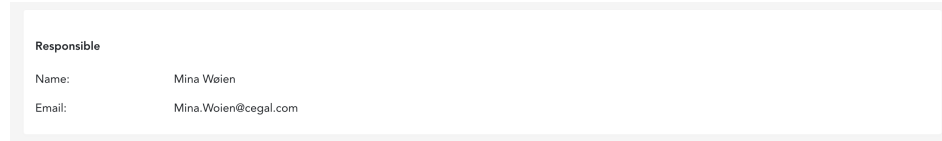


Figure 4.15: Responsible for the engagement on the Engagement information page.

Another section that has been added is the assigned people and resources. This section will either tell you that there are no people or resources assigned, like in figure 4.16. Or it will show all the added consultants, as shown in figure 4.17. When a consultant is added, it will show the user the consultant's name and phone number.

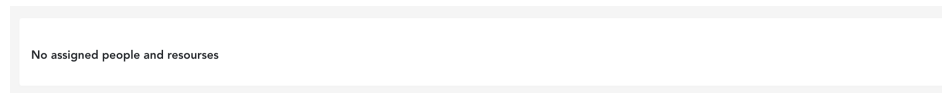


Figure 4.16: No resources assigned to the engagement.

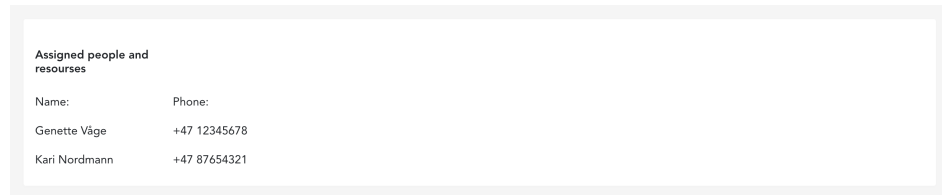


Figure 4.17: Resources assigned to the engagement (Dummy data).

At the bottom of the information page, a details section is placed, as shown in figure 4.18. This section contains a description of the engagement. That will typically be the company's whole description text in their engagement requests. In addition to this, there is a small field called notes. Notes are there to give the user a place to write down important info regarding this engagement.

4.3 Engagement

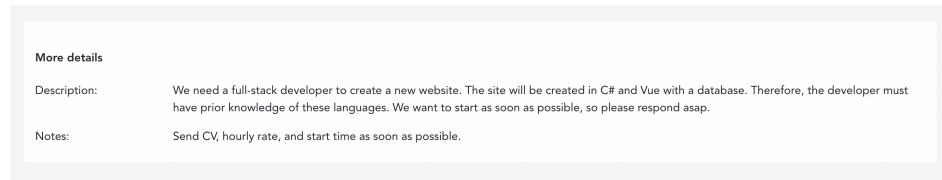


Figure 4.18: The engagement's description and notes (Dummy data).

Since there will always be discrepancies in the application when uploading a program or misspelling when adding an engagement manually, the user can edit the engagement information anytime, as shown in figure 4.19.

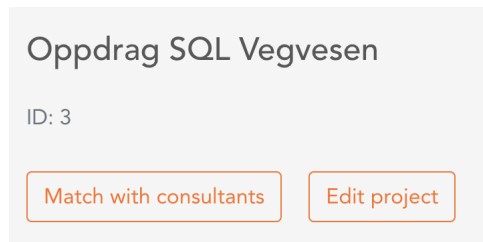


Figure 4.19: Possibility to edit the engagement (Dummy data).

4.3.3 Add new engagement

Because of multiple ways to receive an engagement, there are two ways to upload an engagement, as shown in figure 4.20. You can either upload the engagement manually or upload a file.

4.3 Engagement

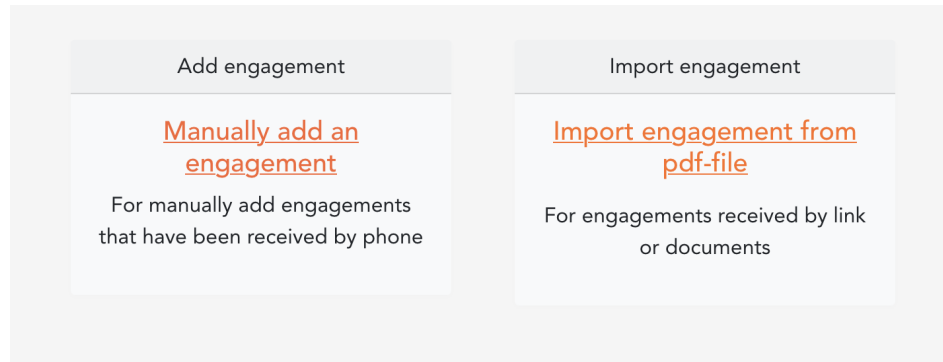


Figure 4.20: Add new engagement page

Manually

When users get an engagement request over a phone call or oral and do not have a file to upload, they can quickly enter the information they got manually, as shown in figure 4.21. However, when the user is going to add an engagement manually, some fields are required. These fields are marked with small red stars to indicate that they are needed. After the user has added all the required information, it will push the “Add engagement” button at the bottom of the page and will be redirected to the engagement overview page.

4.3 Engagement

← Back

Engagement info

Engagement name* Company*

Short description

Deadlines

Deadline for reply Start date* Duration (months)* Optional period (months)

dd.mm.yyyy dd.mm.yyyy 0 0

Requirements

Allocation requirements Education requirements Skills (separate by comma) *

0 %

Other

Location* Type of engagement* Max rate Workplace possibilities

0 Office

Remote

Commute

Resource owner Number of resources* Category Priority

0

Notes

Add engagement

Figure 4.21: Add engagement manually.

File upload

The file upload option is the primary option in this application. However, after attending many meetings and getting a better insight into how Cegal works, there is no doubt that they mostly get engagement per email. These emails either explain the engagement in the actual mail or has a PDF or a Docx attached. The file that is uploaded to the application must be a PDF. The user will be redirected to the engagement overview page when the PDF is uploaded.

4.4 Match

4.4 Match

The match page and match function is the heart of the task. When the user enters the matching page, an overview of all the consultants matched with this engagement will be visible, as shown in figure 4.22. The summary will display the consultants with the highest skills score at the top and will allow the user to decide if they want to sort by skills or other columns.

The screenshot shows the 'Matches' page in the cegal execo system. The page header includes the logo, navigation links (Home, Consultants, Engagements), a 'Logout' button, and the user's name 'Mina Weien'. A 'Back' button is visible on the left. The main content is a table titled 'Matches' with a search bar. The table has columns for Consultant, Skills (sorted ascending), Utilization, Location, Education requirement, and Position. Each row contains a 'Show skills' button, a 'Show more info' button, and an 'Add consultant' button.

Consultant	Skills ↑	Utilization	Location	Education requirement	Position		
[Redacted]	7	81.1%	Stavanger	Not satisfied	Senior Business Consultant	Show more info	Consultant added
[Redacted]	7	0%	Lysaker	Not satisfied	Senior Business Consultant	Show more info	Consultant added
[Redacted]	6	0%	Stavanger	Not satisfied	Business Consultant	Show more info	Add consultant
[Redacted]	5	N/A	Stavanger	Not satisfied	Senior Business Consultant	Show more info	Add consultant
[Redacted]	5	79.5%	Oslo	Not satisfied	IT Consultant	Show more info	Add consultant

Figure 4.22: Match page.

To simplify the display of skills, we have added a “Show skills” button. When the user pushes this button, the consultant’s skills will be displayed, as shown in figure 4.23.

4.4 Match

The screenshot shows the Cegaleco Match page. At the top, there is a navigation bar with the Cegaleco logo, links for Home, Consultants, and Engagements, a Logout button, and a user profile for Mina Woien. Below the navigation bar is a Back button. The main content area is titled 'Matches' and features a search bar. A table displays two consultant matches. The first match has a consultant name redacted, 81.1% utilization, is located in Stavanger, has an education requirement of 'Not satisfied', and is a Senior Business Consultant. The second match has a consultant name redacted, 0% utilization, is located in Lysaker, has an education requirement of 'Not satisfied', and is a Senior Business Consultant. Both matches have 'Show more info' and 'Consultant added' buttons. The first match's skills are listed as: microsoft office, sharepoint, excel, power bi, microsoft azure, sql, and collaboration. The second match's skills are listed as: 7.

Consultant	Skills ↑	Utilization	Location	Education requirement	Position		
[Redacted]	microsoft office sharepoint excel power bi microsoft azure sql collaboration Hide skills	81.1%	Stavanger	Not satisfied	Senior Business Consultant	Show more info	Consultant added
[Redacted]	7 Show skills	0%	Lysaker	Not satisfied	Senior Business Consultant	Show more info	Consultant added

Figure 4.23: Match page where skills are shown.

The utilization indicates how much the consultant is allocated to other engagements in this period. So, for example, if the consultant is 100% utilization, they do not have spare time for this engagement. If this consultant is needed in this engagement, the user knows there has to be a reorganization of consultants.

The overview also shows the consultant's location and whether the consultant has an education that satisfies the engagement requirements and the consultant's position.

On the right side on the match page, as shown in figure 4.23, there are two buttons. The first button will redirect the user to the consultant information page, and the other button will add the consultant to the engagement. After the user has added the consultant to the engagement, it will show a text saying "Consultant added".

Chapter 5

Discussion

This chapter will discuss what has been challenging through the development of the program, including the program's speed and further work. It will also contain the company's evaluation of the program.

5.1 Environmental accounts

The resource management tool is made to increase Cegals income and productivity. There is a potential of increasing the utilization of consultants, and shortening the sales cycle for new hires and returning consultants.

For example, one "filtered" and relevant customer request for a consultant leads to an email written and sent to three to five key stakeholders, spending 10-20 minutes each reading, processing, and responding. There might be a lot of filtered and relevant customer requests across the company per month. By introducing the resource management tool to Cegal, they will most likely be able to halve the time they spend on these customer requests. That in itself is a considerable cost reduction.

Not having consultants in projects is an undesired value leakage for Cegal. If they do not start early enough, it is assumed it takes some weeks to find a new assignment for a returning consultant. This means the consultant

5.2 API

will not have a project after finishing the current project. Therefore, Cegal wants to make sure consultants avoid being “between projects” for two months. It is generally hard to find billable assignments that cover the residual percentage of a billable assignment less than 100% allocated. The resource management tool will help Cegal minimize the number of consultants that end up being between projects, and this will help Cegal in enabling better revenue and higher productivity.

5.2 API

Due to our lack of experience with external API and poor documentation, it was challenging to get started with the API. Through testing with `python-requests`, we validated a single request to Resource Guru. Once we had something that worked, we quickly managed to automate it. However, since tokens need to be revalidated every week, we selected to store the token refresh information in a JSON file. The program reads the JSON file before an API call is done and creates a header with the access token. The access token needs to be refreshed if expired, which is currently done manually.

5.3 Performance of the application

While developing the program, it has been observed that some functions are time-consuming. With Python’s performance counter, we could calculate the time of the different functions.

5.3.1 API calls

The most time-consuming functions are the API calls to CV Partner and Resource Guru when all the consultants are received. The program uses almost 3.5 minutes on the update function that retrieves the consultants automatically, as it gets all of the several hundred consultants from Resource Guru.

5.3 Performance of the application

The CV Partner API gathers the user information about each consultant and includes an individual ID to access each user's CV. Therefore, the program sends two API calls to CV Partner for each consultant, as explained in chapter 3.2.2. The information is provided as a dictionary containing several dictionaries and lists. The data had to be structured before being stored in the database, as only the required data is stored. As the information in each CV comes in a large data format, with lists inside lists, it was hard to find a way to make the code faster.

For example, the function that collects the skills from previous projects. To get these skills and store them in the database, first, the program loops through each consultant. Second, it loops through their project experiences, and finally, through skills before it can check whether the consultant has written their skill in English or Norwegian. This looping made the function a runtime of $O(n^3)$. A runtime with high O-notation for large data sets makes the program slow [16].

There are around 800 people at Cegal's CV Partner and at Resource Guru, and to get 10 of their CVs from CV Partner took almost 7.8 seconds.

On the other hand, getting 28 consultants and their information from the database takes only 0.1390 seconds. This is why we needed a table for the consultants in the database, as explained in chapter 2.2. Then the consultants could be retrieved from the database instead of retrieving them with API calls every time they are needed.

5.3.2 Match function

Another time-consuming function is the match function. Some of this code is from the previous bachelor thesis. They have used multiple for-loops. To match an engagement with consultants when there are only ten consultants in the database takes almost 3 seconds. When the number of consultants in the database increases to 800, multiple for-loops will be more time-consuming.

The goal for this task is that assigning consultants to engagements should go faster. However, even though the match function takes more time than wanted, the program makes the allocation process more efficient. Before,

5.4 Further Work

the matching process had to be done manually between the engagement, Resource Guru, and CV partner, as explained in chapter 2.3.2.

5.3.3 Database indexing

A way to increase the speed of the code is by using database indexing. However, in this application, the consultants get updated daily, and it is not recommended to use indexing if the data gets often updated [20].

5.4 Further Work

While creating the resource management tool, the work has been introduced to different people who will use the program at Cegal. They have come up with various suggestions for changes and features they would like and can benefit from. Due to the time limit and the importance of having a working MVP in the end, some parts were prioritized. However, if the program would be further developed or there was more time, the wishes for extra features were written in the backlog.

5.4.1 Evaluating consultants

In one of Cegal's weekly resource and engagement meetings, they needed to find consultants with a particular skill or experience. A new feature could manage this problem by adding a search function for skills on the consultants' overview page. Then it would probably be relevant to list all consultants that match the skill and their availability.

Another possibility would be to show the skills the engagement requests on the matching page, and be able to filter based on these. Then, only the consultants that match the chosen skills would be listed. To integrate this into the program, we would need a function that searches for skills in the PDF files. The uploaded files match with consultants based on the whole file, not skills from the file. A way to find the skills is to have a list of common skills based on all the consultant's skills from CV Partner,

5.4 Further Work

which could be extended if new skills were added. Then the skills in this list could be searched for in the uploaded file, and the skills found could be added as the engagements wanted skills. If each engagement had preferred skills, a present match could also be added based on skills, as we planned to implement at the beginning of the project.

At one of the meetings with the Sales department, a feature was requested. The request was to be able to upload the CV for the consultants that have not yet started, the so-called no-names. Now they have to be added manually. However, this requires access to all new employees' CVs. It would have been possible with more time and the required permit. The CVs could be handled as the uploaded engagements are handled. Then, the program could search for the different categories needed, and the users could be able to edit the information received from the file.

Another thing this person from Sales requested was an average rate on consultants, as some engagements do not require a senior consultant. If an average rate for a consultant were added, the company that demands a consultant might not be willing to pay the rate set for the consultant. As explained in chapter 3.5, the position of each consultant was added to help the user decide what type of resources they will use for that specific engagement. However, they suggested making a function that calculates the average rate based on their position, such as a trainee, senior, etc. This calculating function is a helpful feature that would be prioritized with enough time. A list with the average rate for the different positions would also be needed.

At another project presentation, a person was missing some of the categories for the engagements. Therefore, he requested that the program could have a link to Salesforce, where all the categories for Cegal's sales are located. To get this link between our program and Salesforce, we would have to have the proper access to this site and include another API in the program.

There was also a request for a priority option on the match function. Then the resource owners could prioritize a previous engagement similar to the requested engagement. In the program now, the users have to view the matched consultant's profile to see their previous engagements. If an input field was added to write in the name of a similar engagement, the match function could sort out the people on this project. However, another way is

5.4 Further Work

to add previous engagement to the matching page and let the user search for it.

5.4.2 Resource Guru API

As the access to Cegal's Resource Guru account was not given to the application before the end of the project, as explained in chapter 5.2, it was not time to do the optimal improvements for the program's speed.

All the resources are retrieved with the Resource Guru API every time the update function runs. Then, another API call retrieves the email address for each resource and updates the utilization ID in the database, as explained in chapter 3.2.3. If there were more time, a function would be created to check whether the utilization ID is added to the consultant table in the database or not. The function would avoid the last-mentioned API call for the consultants already added to the database and make the update function quicker.

After the access token has expired, it has to be manually refreshed. The access token received is short-lived, and the refresh token is long-lived [6]. The refresh token is the token we would use to refresh the access token.

5.4.3 Unassign consultants from engagements

As the users can assign consultants to engagement, they should have been able to unassign them. Since it is easy to make a mistake or other users may disagree with the user who assigned the consultant to an engagement. For better user-friendliness, further work would be to unassign them from engagements. A delete button could be added on the engagement and consultant pages, which deletes the relationship between the engagement and consultant in the database. An unassign button could also be added to the matching page where the consultants get added as the users easily can undo their assigning if they clicked in error.

5.4 Further Work

5.4.4 Data security

Security is more important than ever and especially when personal information is involved. Only a small group of people will have access to the site, and no one else must get their hands on this information. Therefore, our priority in our future work would be to develop further and strengthen security.

For example, to avoid the user's profile being abused or the user's session being hijacked, an automatic logout function that logs the user out after a specific time could be implemented [21].

Since Azure AD is used for log-in, this logout function has to be configured in the Azure Portal. The web app session timeout is set to "rolling" as default [12], which means that the user is stored in the cookie every time the user refreshes the site unless the user logs out. By changing the web app session timeout to "absolute", we could set a time for when the user has to re-authenticate.

Bibliography

- [1] . Azuread. <https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis>.
- [2] Flow charts, identify and communicate your optimal process. https://www.mindtools.com/pages/article/newTMC_97.htm.
- [3] . Uploading files - flask documentation (2.0.x). <https://flask.palletsprojects.com/en/2.0.x/patterns/fileuploads/>.
- [4] What is a backlog? <https://www.productplan.com/glossary/backlog/>.
- [5] What is a data model? <https://www.erwin.com/solutions/data-modeling/data-model.aspx>.
- [6] Daniil Penkin. Refresh tokens using python requests. <https://community.atlassian.com/t5/Bitbucket-questions/Refresh-Tokens-using-Python-requests/qaq-p/1213162>, 28.okt 2019.
- [7] Claire Drumond. What is scrum? <https://www.atlassian.com/agile/scrum>.
- [8] Høyvik, S. and Grimstad Lierstuen, J. Socmo - smart optimal consultant matching and offering. [Unpublished bacehlor's thesis]. University of Agder, 2021.
- [9] Intersoft Consulting. Security of processing. <https://gdpr-info.eu/art-32-gdpr//>.
- [10] KirstenS. Cross site scripting (xss). <https://owasp.org/www-community/attacks/xss/>, 18.sep 2015.

BIBLIOGRAPHY

- [11] Microsoft. Manage secrets in your server apps with azure key vault. <https://docs.microsoft.com/en-us/learn/modules/manage-secrets-with-azure-key-vault/>.
- [12] Microsoft. Configure session behavior in azure active directory b2c. <https://docs.microsoft.com/en-us/azure/active-directory-b2c/session-behavior?pivot=b2c-user-flow>, 14.mar 2022.
- [13] Microsoft. Best practices for secure applications. <https://docs.microsoft.com/en-us/linkedin/shared/api-guide/best-practices/secure-applications>, 4.jun 2022.
- [14] NTNU. Two-factor authentication. <https://i.ntnu.no/wiki/-/wiki/English/Two-factor+authentication>.
- [15] Python. multiprocessing — process-based parallelism. <https://docs.python.org/3/library/multiprocessing.html>.
- [16] Semi Koen. The big o notation. <https://towardsdatascience.com/the-big-o-notation-d35d52f38134>, 19.jun 2020.
- [17] Stackoverflow. Check whether a pdf-file is valid with python. <https://stackoverflow.com/questions/559096/check-whether-a-pdf-file-is-valid-with-python>, 18.sep 2015.
- [18] SuperOffice Norge AS. Hva er gdpr, og hva betyr det for din bedrift? <https://www.superoffice.no/ressurser/artikler/hva-er-gdpr/>.
- [19] The PostgreSQL Global Development Group. PostgreSQL: The world's most advanced open source relational database. <https://www.postgresql.org/>.
- [20] Wikipedia. Database index. https://en.wikipedia.org/wiki/Database_index, 7.jan 2022.
- [21] Åse J. Sagebakken and Liv Underhaug and Mina Wøien. Dat250 - report. [Unpublished DAT250 report]. University of Stavanger, 15.apr 2020.