Cieve Software Engineering Report

Felix Gaul, Nathan Hall, Josh Hankins, Abdullah Muhammad-Kamal, Alistair Robinson, Justas Tamulis

Group 24

Preface

Through our understanding of Deutsche Bank's problem statement, our research of existing systems available in the market and a direct consultation with the client, we have developed a streamlined, intelligent recruitment system capable of using machine learning to handle the applicant screening process for our client: Cieve. This document describes our journey through the development cycle of Cieve, from the formation of its development team through to our planned development and support roadmap for the future.

Our source code can be downloaded from
https://drive.google.com/file/d/16BK3Or_A52l9t003LIHdxQvedMjeMW0K/view?usp=sharing

Introduction

The 2019 DBCampus Project was to develop a prototype smart application system for use by Deutsche Bank. This task was given to all second year Computer Science students at the University of Warwick with the intention of pushing students software development boundaries, in a business environment. We, Team Cieve, believe that we have developed a recruitment system more than capable of not only meeting, but surpassing the requirements given by our client. Furthermore, we have discussed the additional functionality and considerations we have made on top of the client's initial requirements to differentiate ourselves in the market for smart application systems.

This document begins by outlining our Research and Design process for the application, which is discussed further in our associated *Requirements Analysis* and *Planning and Design* documents. The process of implementing these areas of the project is described in our Implementation section. Our team also undertook the significant task of ensuring that the application is fully compliant with modern data protection regulations, such as GDPR, and that the system is also protected against many common security vulnerabilities. This process is detailed in the Security and Compliance sections. Our approach towards Test Driven Development is then explained. This is followed by our evaluative process, outlining any unsatisfied requirements, application maintenance and areas identified for future development.

Research

We decided it would be effective to research the systems currently in use by the client and competitors.

Client Systems

In creating the online application system for the client, it was important to first become familiar with the firm's philosophy. We inferred from our interactions with the client through guest lectures and our client consultation that incoming applicants were more likely to be successful if their behaviours aligned closely with that of their potential employer. We began our research process by studying Deutsche Bank's current application system in depth, going through the application process to identify areas that could be made more streamlined or intelligent by our system, which were then associated with requirements in our *Requirements Analysis*. A full report of our findings is available in Appendix [1.1].

Competitor Systems

We also preformed research into existing smart application systems, including how we felt that we could improve on each system. A full report on these findings is available in Appendix [1.2]. Performing research into these systems helped us to identify the exact requirements necessary to differentiate ourselves from other competitors in the smart recruitment system market.

Machine Learning

Our machine learning approach was selected using prior knowledge from an online training course in machine learning systems written by Stanford University[1]. This introduced us to neural networks, backpropagation and gradient descent, which we adapted to suit given task. We are using sparse neural network with batch gradient descent as an error minimization function. Third party libraries were not used in our implementation of machine learning.

Technologies Used

Finally, to assist us in our development process for the application, we decided to research frameworks and existing technologies which could assist us through our development process. Our preliminary research and existing knowledge of the recruitment sector informed us that a platform agnostic web application would be the most effective deliverable to the client. We would therefore require:

- A web server framework (Python Flask, due to flexibility and team experience with Python[2])
- A database management system (MongoDB Atlas, due to scalability and reliability[3])
- A unit testing framework (Pytest, due to its use in industry and base compatibility with Flask[4])
- A website frontend framework (Bulma, due to lightweight design and mobile support[5])

[1] www.coursera.org/learn/machine-learning/home/welcome
[2] www.flask.pocoo.org
[3] www.mongodb.com/cloud/atlas
[4] www.docs.pytest.org
[5] www.bulma.io/

Design

The design of the application began after completing our preliminary research, deciding software development methodology and familiarising ourselves with our chosen development technologies. Our initial design concept can be found in our *Planning and Design* document. Throughout our development period we made several decisions to improve, adapt or remove certain aspects of our design.

Data Model

Our database schema changed during our development process to facilitate the inclusion of additional features, such as our phishing codes and user messages. Additional collections included the *interviewStage* and *questionStage* tables, which stored data specific to user testing data. We also added a *metaData* collection to store new input data such as new skills. We separated login details for applicants to ensure data protection compliance, allowing us to preserve user accounts whilst deleting expired applications after six months of storage in the database. This affected our data model and class model, however, the magnitude of changes made were not considered sufficient for us to redesign our entity relationship and class model diagrams.

Applicant Processing

We decided against using a third party analysis tool to process applicant data, instead using our own bespoke algorithm to score applicants based on their submitted information, vacancy information and a stage specific score. We did, however, make minor adjustments to our evaluation model:

- Scores are now calculated using sparse neural network, which uses the batch gradient descent machine learning algorithm to continuously improve its accuracy.
- We receive feedback from the client whenever a job is closed or when they edit a machine learning weighting in their dashboard. Any accepted candidates have a positive influence on weights, rejected candidates have a negative influence on weights.
- Batch gradient descent was selected as core of our machine learning algorithm. We used heuristic algorithms on the sample data provided to train our model to identify higher scoring candidates.

User Interface

We made minor adjustments to our user interface design to accommodate pages for new functionality, such as a machine learning dashboard. We also adapted our designs to be encourage accessibility on mobile devices. This can be seen in Appendix [2.1]. We adapted designs to shorten and streamline user journeys on all devices. Our wireframe developed in AdobeXD was fully translated into templates for Python Flask web pages.

Implementation

User Interface

We chose to implement a minimalist user interface (UI), so as not to confuse applicants and clients. Page design was constructed using 'mobile first' development to appear more mobile friendly and used a simple colour scheme which was accessible to colourblind and neurodiverse users. The following pages were developed by our frontend team using the Bulma framework alongside jQuery, which was chosen as our Javascript framework due to its support for asynchronous Javascript (AJAX):

```
 - / applicant pages                 - / client pages
 | - apl/                            | - cli/
   | - auth/                           | - auth/
     | - login                           | - login
     | - register                        | - register
   | - jobsearch                       | - jobs
   | - newapplication                  | - jobBreakdown
   | - applications                    | - stageDetail
   | - testing                         | - newJob
   | - booking
```

From the client side, we allow recruiters to create and advertise jobs via the *New Job* page. We allow clients to specify preferred skills and development languages to allow our machine learning subsystem to generate job specific scores for applicants. Furthermore, clients can intelligently personalise a vacancy's recruitment process using our stage builder, where each stage is either an interview (e.g. phone interview) or an online test which users must complete (e.g. logic and reasoning tests). This vacancy can be viewed alongside others made by the same client using the *Your Jobs* page, where a client can access a job breakdown of all applicants and their current stages, sorted by either their job score, or overall score generated by the machine learning subsystem and progress them through the application process.

From the applicant side, we allow users to search through all jobs listed in the current database via the *Job Search* page. We use a separate page to take applications from users, for which they can specify a set of preferred vacancies and opt into additional consideration for new jobs in the same department, role and location. This page sends back useful information from the client to the server for use in the machine learning subsystem via an AJAX request. An applicant can view the status of their application through the *Applications* page. Once an applicant has been progressed to an interview or testing stage, they can either complete tests on the *Testing* page or book an interview slot using the *Booking* page. Results from *Testing* are fed back into our machine learning subsystem to calculate the user's stage score and booked interviews are fed back to the recruiter. Once the applicant has passed through all necessary stages, a client can then make a job offer and hire them into the business, completing the recruitment process.

The usability of our UI was validated through user acceptance testing, where testers from a wide range of technical skill levels, English fluency levels and neurodiversity backgrounds were timed and asked to complete tasks on the site. Feedback was integrated back into the UI to improve usability for all users.

Web Server

Our chosen web server framework, Python Flask, was crucial to the success of our application. Flask's flexibility and security functionality allowed us to logically and methodically build a server to handle the bulk of application processing in our system.

We began by implementing an authentication framework for the system to enforce strict access across the application. From this, the next priority of the web server development team was client and applicant functionality, including vacancy and application functionality. The team was then required to integrate the machine learning subsystem into the web server, so that new applications would be scored as per the client's specification. The final stage of web server development was the completion of additional functionality on top of the requirements given to us by the client in their problem statement, including but not limited to: variable staging in vacancy creation, interview tracking and booking, online user testing, general applications. These distinct sections of the project were constructed as Flask blueprints, improving our system's portability and effectively allowing different subsystems of the application (e.g. applicant functionality, client functionality) to be hosted on two different machines if necessary. Throughout the development of the web server, our Pytest unit testing framework was expanded to include validation tests for new functionality, through both white and black box testing.

After completing any functionality in the web server, the frontend was instructed to begin work on the HTML template and user interface which would allow clients and applicants to utilise the newly implemented page. This worked in conjunction with the verification of the web server's unit testing framework to allow team members to validate functionality against customer requirements.

One exchange involving all subsystems is when an applicant submits an application to a vacancy. From the frontend, AJAX is used to reduce the perceived processing time for the request so that the user does not have to wait for the machine learning to complete processing to continue using the website. The data is caught by a webhook on the server, `/newapplication`, where the json formatted data is translated into a dictionary understood by the machine learning subsystem. Once error checks pass, the application data is sent to the machine learning evaluator, which assesses the applicant to generate a basic score and stores the result in the database. Once this is complete, the evaluator then generates a job score for each vacancy that the user identified and stores these values in the database. This process does not affect other users' experience of the application since our machine learning is designed to be minimally computationally expensive and the Flask framework is multithreaded as standard.

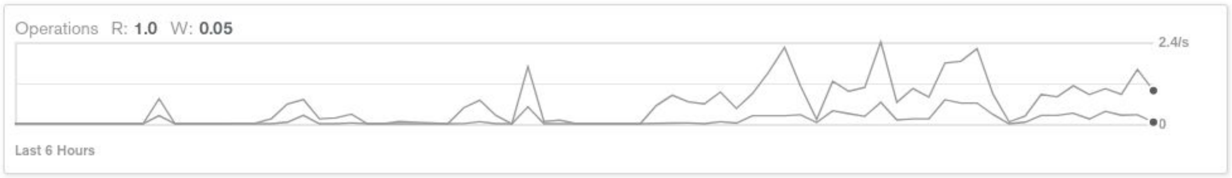
Following our research into possible web server functionality, we elected to host our application locally. Whilst this limited our availability to external users, this worked well with our cloud hosted database management system and encouraged us to design our application to be platform agnostic to improve portability following our research into Capita's British Army recruitment system[6]. It also fits within our budget constraints. Our application was developed on various operating systems and is also designed to be compatible with Python virtual environments, further adding to our system's portability.

The web server contained the most third party dependencies of any section of the project, due to its size and complexity. Dependencies were listed in the application's *requirements.txt* for GitHub's vulnerability tracker (see *Security*) and Travis' Continuous Integration platform (see *Test Driven Development*).

[6] www.nao.org.uk/wp-content/uploads/2018/12/Investigation-into-the-British-Army-Recruiting-Partnering-Project.pdf

Database

Committing to our focus on portability, reliability, security and scalability, we chose to host our MongoDB database on a cloud server, provided as a server by MongoDB through their free Atlas service. This allowed us to remotely connect to and perform Create, Read, Update and Delete (CRUD) actions on a database cluster without cost to our budget. From a portability standpoint, this meant that any host would be able to access the database without the necessary requirement to be hosting a MongoDB database (e.g. low spec devices, outdated devices). This also helped us during development since we could rely on the data in our database being the same regardless of access point. From a reliability standpoint, MongoDB Atlas can automatically take backups of premium databases[7] to allow rollbacks in the event of data loss. MongoDB Atlas also has a Service Level Agreement uptime of 99.995%[8] across its servers, and if one server fails the cluster is automatically migrated to another server to reduce disruption. From a security standpoint, MongoDB Atlas has undergone extensive security checks, enforces SSL encryption for all database communications, blocks NoSQL injection and automatically logs transactions on the cluster. From a scalability standpoint, the cluster's capacity, both in terms of storage and memory, can be increased to adapt to future increased loads. These factors caused us to favour a cloud based database rather than a local database. MongoDB Atlas also allowed us to view the efficiency of our application's database interactions, which would help us identify points of high traffic after deployment:



In order to access the database, a `db.py` file was created in our database to act as an interface between the backend subsystems and the cloud database. To facilitate this, we used the `pymongo` third party database driver to establish a link to the cloud database. The database class was responsible for performing CRUD actions on our remote database to fulfil functional requirements, such as retrieving an applicant's information from their ID or username. In addition to this, our database file contained a function to automatically find non-compliant data (e.g. expired applications) and delete entries before they can be accessed. This function was used whenever retrieving personal data from the database. A file containing insertion commands for initialisation data was also created in `db_setup.py` to populate the database with its constant metadata (collections `metaData` and `feedbackWeights`) alongside useful sample data which was used by team members to perform validation testing with new features.

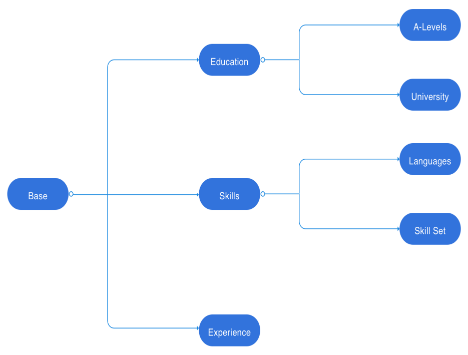
One difficulty identified during implementation which would not have been apparent in an SQL based database was the lack of foreign key references and automatic cascading when deleting the data. This meant that cascading deletion had to be programmed manually and required extensive testing to verify correctness. We were able to define auxiliary cascade functions to reduce code duplication.

[7] www.docs.atlas.mongodb.com/backup-cluster/
[8] www.mongodb.com/cloud/atlas/faq

Machine Learning

Our machine learning subsystem's implementation was encapsulated within an evaluator class in the application's source code. The model uses a machine learning algorithm with a batch gradient descent as optimization algorithm which utilises backpropagation to adjust weights in a sparse neural network to learn from recruiter feedback. This allows us to score users with a basic score (taken from the applicant's input data such as education and employment history), moreover we calculate job specific score by comparing an applicant against the requirements of a specific vacancy. Recruiters can use this information to assist in their recruitment process. A custom sigmoid function was selected as activation function to ensure that applicant scores are distributed between 0 and 1[9] with a high standard deviation to utilize all range of possible score. Weights inside the model are intialised by using a heuristic model to label data and are stored in a collection on the MongoDB Atlas remote database. Recruiters automatically give feedback to our machine learning subsystem by accepting or rejecting applicants, which is fed to subsystem when closing job vacancy postings. Recruiters can also edit weights through the dedicated machine learning control dashboard. The learning rate of the model is directly influenced by the ratio of accepted to rejected applications to help normalise the model to maintain an average score of around 0.5. The diagram below shows a tree diagram representing the calculation of scores by the subsystem. All A-levels, universities, languages, skills, employers and job roles present in the database are associated with nodes in our sparse neural network.

When a user applies to a vacancy, through either a direct or general application route, their inputted data is sent to the machine learning evaluator. The evaluator first calculates the applicant's basic, vacancy independent score according to the scores they receive for their education (degree qualification, degree level, university attended, A-Level subjects and grades), skills (languages known and listed employable skills) and previous employment (companies, positions and length of employment). Once this score has been calculated, the model then recalculates a vacancy specific role for all roles that the applicant is under consideration for. For each job, the applicant receives a penalty for any skill, development language or educational requirement listed by the job which the applicant does not possess. This is displayed alongside the user's basic score in each recruiter's job breakdown to inform recruiters on the suitability of an applicant for both the specific job and the business.



In order to train our sparse neural network, we used heuristic functions on the sample applicant data provided to increase weightings for certain candidate features, such as prioritising universities higher in published league tables. This improved our model's ability to identify better suited candidates.

One major challenge faced during the implementation of our evaluator was how to effectively pass non numeric user data into the model. We mapped qualitative data, such as a university name, to a numeric value in our model using the database. This facilitated allowing users to introduce new variables into the machine learning model. For example, if an applicant inputs a new skill which has not previously been registered by the model, the subsystem adds it to the model with a default weight and updates the database to reflect this.

[9] www.towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Security and Compliance

Our team decided that security should be a consideration throughout all stages of development for our software engineering project to ensure that the final product would not be vulnerable to attacks from malicious parties. Whilst no system can ever be 100% secure, by understanding the risks to our system we were able to significantly improve the security of our application.

Information Risk Assessment

The OWASP Top Ten, a consensus of the ten most critical web application security vulnerabilities, was used as a guide for vulnerabilities that the application should be protected against. A risk register was developed to track potential security vulnerabilities, sorted from high risk to low risk, along with the mitigations taken to reduce the risk of the vulnerability being exploited. This can be found in Appendix [4.1]. Residual risks have been identified in this report's Evaluation section. By tracking and understanding these risks, we were able to effectively treat (add mitigations against), terminate (remove functionality), tolerate (accept the risk) or transfer (move the risk onto another party) the risks to deliver a secure application.

Dependency Security

In order to ensure that our imported third party packages did not introduce additional risk to the application, we used GitHub's dependency security scanner. This service continuously identifies all imported packages used in our repository and checks them against a vulnerability database. If a dependency vulnerability is found, the owner of the repository is notified immediately so that they can update the dependency to a more secure version. This allowed us scaleable dependency vulnerability detection for future development on the project.

External Security

It was not only necessary for our system and its dependencies to be secured against attacks, but also for the frameworks and technologies used to support the project. We read security documentation for each technology used to verify that they did not add an additional risk to our system for all third party technologies used as a part of our development process.

GDPR

With the recent introduction of stricter data protection laws through the European Union's (EU) General Data Protection Regulation (GDPR), the team has ensured that the final product would be fully GDPR compliant by managing our data and obtaining user consent. The UK Data Protection Act (DPA) 2018 was also taken into consideration. Under the European Union Withdrawal Act 2018, GDPR will be incorporated directly into UK law immediately after the UK exits the European Union.

Data Compliance

In our system, applicants are our primary data subjects, users who can be identified through the personal data provided to our service. Clients are our primary data controllers, users or organisations that determine the purpose of collecting data from a set of data subjects Data controllers are responsible for protecting candidate data and using it in compliance with the law. Our applicant machine learning subsystem is our primary data processor. Data processors are systems or users which process data collected from data subjects. Sub-processors can include systems such as MongoDB Atlas, which is used to store user information. Compliance for our system was tracked by identifying the relevant articles of GDPR which affected the application and matching them against any considerations or actions made to enforce compliance in our system. A full compliance report can be found in Appendix [3.1]. Any articles omitted from this report were not considered relevant to the application.

To be fully compliant, the data protection officer was required to define a privacy policy for the system. An open source privacy policy template was used and modified to reflect the data protection policy of the application due to time constraints. Users are required to accept the privacy policy.

Data Storage

Data in motion is any data which is in the process of being transferred, collected or processed by a data processor. It should be encrypted to prevent breaches of confidentiality or integrity as stated in *Article 5*. Due to the timescale of the project, it was infeasible for us to receive a full digital certificate from a certificate authority, so we enforced encrypted communications by reconfiguring our Python Flask web server to establish a HTTPS connection with the user's browser. This means that data in motion between users and the web server is fully encrypted to industry standards. We tested this using Wireshark to intercept data in motion[10]. Data in motion between the web server and the remote hosted MongoDB Atlas database was also encrypted, as per the security configuration of our cluster.

Data at rest is encrypted and protected via the SCRAM authentication system and IP whitelisting[11] to enforce integrity, confidentiality and authentication, as per Atlas' security policy. Furthermore, Atlas automatically logs any read or write requests to the database, satisfying non repudiation and *Article 27* of GDPR. Additional security considerations can be found later in the report. We ensure that expired applications are deleted in compliance with Article 13 by performing a sweep of the database to delete any outdated personal data before we retrieve such data. Unit tests were used to verify this.

Data Processing

Our automated profiling policy is clearly outlined in our privacy policy[12]. Our employee dashboard transparently allows clients to view and modify machine learning weightings. Furthermore, our automated profiling process only ranks candidates for a specific vacancy, so the final decision on whether or not an applicant is successful is made by a human client, avoiding conflict with *Article 22*.

[10] https://www.wireshark.org/
[11] www.mongodb.com/cloud/trust
[12] www.cieve.com/privacy

Test Driven Development

Following Deutsche Bank's guest lecture on how they utilise test driven development in their technology divisions, we decided to use a unit testing framework throughout development as part of our dependability engineering mindset. This additionally allowed us to perform quantitative success measurement near the end of the development process.

Unit Testing

As stated in our initial documents, Pytest was chosen as the testing framework for the program. These tests were fully comprehensive because they checked the expected outcome of internal functions (white box testing) and also simulated HTTP requests and user sessions from outside the application (black box testing). Unit tests were separated into distinct files, each with their own testing scope. Each test consisted of documentation and an associated requirement number if applicable. Our final framework contained 75 unit tests for compliance, requirements, security and availability testing. Unit testing allowed us to catch errors throughout development, both in terms of validity of our code and requirements validation. It was also used to perform success measurement at the end of the project.

When using parallel version control through the use of different branches in Git, it became difficult for us to verify the integrity of our branches before merging with master. Therefore, we utilised a Continuous Integration (CI) platform, Travis CI[13], to verify each concurrent version of code throughout the development cycle. Travis runs pushed code, taken from each available branch, in a virtual machine to ensure that it works correctly. If a build passes, the push is verified. We integrated this CI tool into our `README.md` file to make it easy for all team members to see the progress of their builds. During the later stages of development, our unit testing framework was integrated into our Travis CI script to allow the platform to automatically perform unit tests whenever code changes were uploaded to the repository. This allowed us to pinpoint when the introduction or change of a feature affected previously working features, such as job applications.

User Testing

We decided to also perform user testing to ensure that our system's application process was accessible. We asked a number of testers from a wide range of technical skill levels, English fluency levels, ages, employment and neurodiversity backgrounds to complete a series of tasks on the site. Responses were timed to ensure that tasks did not take longer than they should and testers were asked to express their thought process when performing a task, completing a series of user journeys. Any areas which were ambiguous or difficult to use when performing the tasks were identified. Feedback was correlated against Neilson's usability principles and given to our frontend sub-team to correct for. This led to a number of improvements to our user interface, from correct labelling of sections of the vacancy application process, to a clearer password complexity policy.

[13] www.travis-ci.com

Project Management

Team Structure

It was decided early on that our software engineering team should specialise into different areas whilst utilising pair programming techniques to ensure that each aspect of the deliverable was fully understood by at least two team members. With this in mind, we separated into three sub-teams with each team member being assigned individual responsibilities. This is available as Appendix [5.1].

Due to the nature of the project, we were not able to have much interaction with the client other than the client consultation. Therefore, we maximised our utility of the client consultation by clarifying aspects of the problem statements and verifying our proposed requirements analysis during the short meeting. We kept in regular contact our designated tutor, Marcus King, to clarify that our progress was sufficient.

Version Control

To allow for concurrent development between development teams, we utilised Git version control software. Git allows software to be developed in parallel through the use of branches, which can be developed independently and merged periodically to allow different team members to work on different sections of the project. We used a cloud based Git hosting platform, GitHub, to allow remote access to our Git repository. The use of GitHub also allowed us to use Travis to perform build verification, discussed in the testing section. Three development branches were created to allow each sub-team to work on their sprint cycles without affecting the work of other team members, which would be merged at the end of the sprint cycle if all teams had verified the integrity of their code. This was later replaced with a single shared development branch as the project approached code completeness.

Version control was initially challenging during our development cycle since only two members of our team were familiar with using Git. This meant that all other team members had to familiarise themselves with Git, which led to merge conflicts early in development. However, as development progressed, we found that the frequency of merge conflicts decreased as our team's experienced increase.

Development Approach

As outlined in our design and requirements documents, it was decided that our team should follow the Scrum agile development methodology to allow us to quickly adapt to change over our short 9 week development period. Our sprint cycles (from Week 5 to Week 9) had a length of 3-4 days, allowing for two sprint cycles per week. This period was longer than sprint cycles used in industry but reflected the fact that we were not working on the project in the same environment full time. Meetings were scheduled on start/end dates of sprints to evaluate the previous sprint and plan the next phase of development. During the later stages of development, sprint cycles became more frequent to accommodate feature deployment and bug fixing. We endeavoured to reach a point of “zero bug bounce” in the application. We researched Python documentation techniques such as Pydoc, however, we did not feel that this was feasible given the time constraints of the project and the additional features we had planned. Code documentation was still strongly encouraged but not enforced or standardised.

Evaluation

Project Management Retrospective

Overall, team members were well organised and the project was completed in a structured, logical manner. However, we switched from using a GANTT chart to a list of clearly defined sprint cycles to track associated targets and deadlines. Our agile *Scrum* development strategy was not fully compatible with a rigid development plan, so we instead planned our bi-weekly sprint cycles in advance to ensure that the project would be complete before the deadline and modified these plans to adapt to unexpected setbacks, changes in the development process and external events affecting team members. This can be found in Appendix [5.2].

Success Measurement

The unit testing framework allowed us to test how effectively we had fulfilled the requirements outlined in our requirements and design documents. A full table of unit tests for requirements can be found in the requirements analysis document. In our submitted version, our unit testing framework passed consistently on 77/77 unit tests. However, our unit tests did not account for all requirements outlined in the *Requirements Analysis* due to time constraints, however, this could be corrected for given additional development time. The completeness of our project was also verified by our user testing process.

We found that, due to time constraints, we had achieved nearly all of our must requirements, most of our should requirements and some of our could requirements. We have drafted a table of the estimated development time required to add the functionality of unsatisfied requirements to the project:

Requirement	M/C/S	Reasoning	Est. Additional Development Time
R4 - Applicants will be able to upload a PDF CV during an application to prepopulate the form	Could	Unacceptable security risk from allowing file uploads to server, as well as reduced long term scalability due to server storage constraints. An API could be used to process the information, however, this would affect our GDPR compliance for introducing a third party data processor.	N/A, requirement rejected
R20(B) - The system will implement prevention steps for stopping applicants ‘gaming’ the application process	Could	Time constraints meant that we were unable to train our machine learning framework to identify candidates which were lying with sufficiently consistent accuracy.	4 days

Application Maintenance

We have identified areas of the project which may require developer intervention to deliver a consistent level of scalability and dependability in the future:

Aspect	Reason Given	Est. Cost
Failure Recovery	In the event that the server is disabled, a developer will be required to restart the server to resume provision of service.	Low
Database Storage Constraints	Whilst application data is automatically deleted after expiry, high usage may cause the MongoDB Atlas database to reach its storage capacity. A developer will be required to either delete irrelevant data or increase the database cluster's storage capacity.	Low
Cyber Attack	In the event of a successful cyber attack, developers will be required to manually fix the vulnerabilities exploited to perform the attack. The data protection officer will be required to give a statement and report the incident to the Information Commissioner's Office. Business analysts will be required to report the incident to users and clients	High

Future Development

In addition to the unmet requirements already identified, we would like to outline our future development road map for the project. These are any additional requirements which we believe could be feasibly integrated into the project.

R#	Type	Customer Requirement	Developer Requirement	Est. Additional Development Time
R29	Functional	Allow users to receive email updates on the status of their applications and vacancies	Integrate an email server into the application capable of updating when events occur in the main web server	7 days
R30	Functional	Allow users to create and apply to vacancies through a mobile application	Develop an intelligently designed mobile application to interface with our web server's API	7 days
R31	Non Functional	Give clients demographic breakdowns for each vacancy's applications	Collect additional personal data (with permission) and aggregate it into a report for clients	7 days
R32	Functional	Use cookies to remember users to avoid having to login multiple times	Implement browser cookies to reduce friction. Update privacy policy to match.	3 days

Conclusion

We feel that our completed project, Cieve, not only fulfils but exceeds the requirements set out by our client. Our project exceeds in its accessibility and flexibility for all applicants and clients, regardless of profession, without compromising areas such as security, scalability, reliability, fault tolerance and legal compliance, thus representing the intelligent recruitment system that the client's representative described in our consultation. Our unit testing framework has worked in conjunction with our user testing process to ensure that our project meets the requirements outlined by the client's problem statement and our *Requirements Analysis* document. Risks to the project, from the beginning of development through to live deployment, have been tracked, analysed and managed effectively to minimise the residual risk passed onto the client through the integration process. We have discussed areas of maintenance and additional functionality which could be implemented into the project given a longer development period, should the client wish for us to continue developing Cieve into an even stronger application. We look forward to hearing your feedback on our application.

Team Cieve group photo

Appendices

[1.1] Research Report for Client Systems[14]

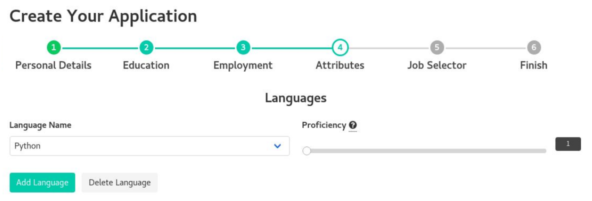
Issue	Justification	Solution	R#
Interview selection process is performed manually	This adds a cost to the client both financially through hiring a recruitment team and through delays by increasing the time lag between posting a job vacancy and having new workers enter a team	Use machine learning to automatically identify stronger candidates	R5, R6, R7, R8, R9, R17
Candidates only apply to one job at a time and are not matched to jobs	Candidates should be able to apply once and be considered for multiple roles across the business to reduce time cost, prioritised for well suited jobs	Allow for general applications to a business, prioritising candidates which match the skills of a candidate	R19
Application progression is static and inflexible	Some vacancies may require alternative progression stages and skills testing	Allow clients to choose their application stages when they create a vacancy	R12

[1.2] Research Report for Competitor Systems

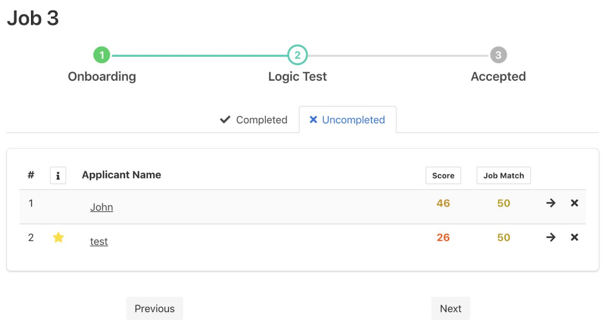
System	Advantages	Disadvantages	Improvement
SniperAI[15]	Utilises machine learning and artificial intelligence effectively to reduce time and financial cost for client businesses	Companies cannot modify machine learning heuristics to match applicants to their corporate culture.[16]	Allow clients to have direct control over machine learning weightings.
Amazon Recruitment AI[17]	Significantly reduced recruitment costs for the business from 2014-2015	Machine learning heuristics began to develop bias against women and project was scrapped[18]	Allow clients to have direct control over machine learning weightings. Do not use protected data in ML.
Pymetrics[19]	Uses custom AI to reduce unconscious bias in recruitment algorithms. Utilises research into neuroscience.	Lack of direct human intervention, computer error may lead to the automatic rejection of a perfect candidate, potentially conflicts with Article 22 of GDPR	Only allow a client to outright reject an applicant, make all applicants available for consideration.
Capita British Army Recruitment system[20]	Designed to increase online applications and reduce costs to the Ministry of Defence (MoD) by reducing the need for physical recruitment centres	Massively over budget, slow feedback. Deployed late due to lack of portability from Capita to MoD systems.[21]	Develop an easily portable system which is as platform agnostic as possible. Give applicants instant feedback on progression.

[14] https://www.db.com/careers/
[15] www.recruitmentsmart.com
[16] www.seedrs.com/recruitmentsmart
[17] www.amazon.jobs/en/
[18] www.theguardian.com/technology/2018/oct/10/amazon-hiring-ai-gender-bias-recruiting-engine
[19] www.pymetrics.com/employers/
[20] www.army.mod.uk/
[21] www.nao.org.uk/wp-content/uploads/2018/12/Investigation-into-the-British-Army-Recruiting-Partnering-Project.pdf

[2.1] User Interface

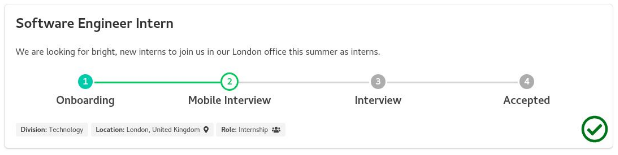


- *Candidate Application Creation*



#	i	Applicant Name	Score	Job Match		
1		John	46	50	→	✕
2	★	test	26	50	→	✕

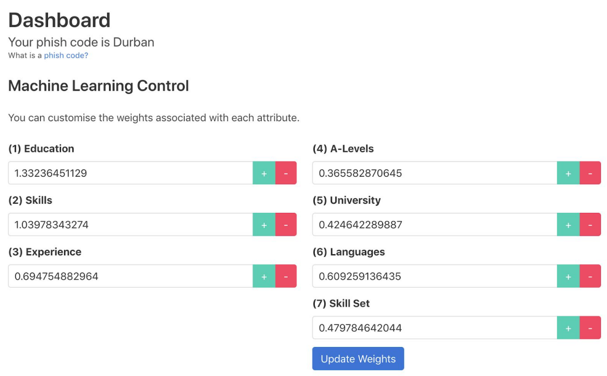
- *Recruiter Application View*


Software Engineer Intern

We are looking for bright, new interns to join us in our London office this summer as interns.

Division: Technology **Location:** London, United Kingdom **Role:** Internship

- *Candidate Applications Summary*


Dashboard

Your phish code is Durban

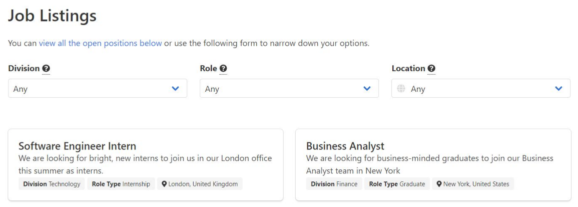
What is a phish code?

Machine Learning Control

You can customise the weights associated with each attribute.

(1) Education: 1.33236451129	(4) A-Levels: 0.365582870645
(2) Skills: 1.03978343274	(5) University: 0.424642289887
(3) Experience: 0.694754882964	(6) Languages: 0.609259136435
	(7) Skill Set: 0.479784642044

- *Recruiter Dashboard*


Job Listings

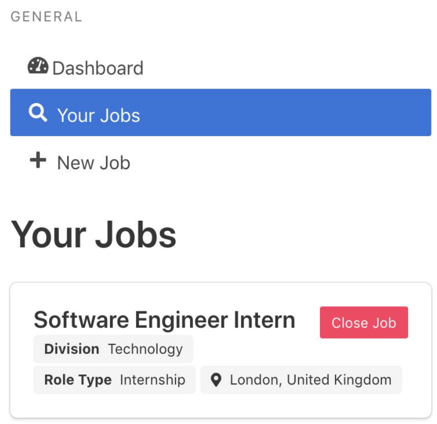
You can view all the open positions below or use the following form to narrow down your options.

Division	Role	Location
Any	Any	Any

Software Engineer Intern
We are looking for bright, new interns to join us in our London office this summer as interns.
Division Technology **Role Type** Internship London, United Kingdom

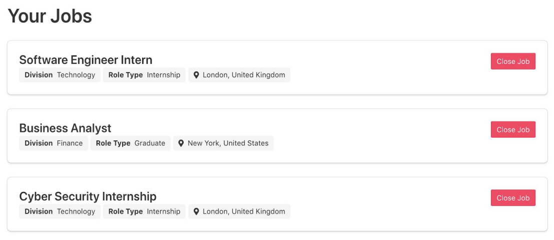
Business Analyst
We are looking for business-minded graduates to join our Business Analyst team in New York
Division Finance **Role Type** Graduate New York, United States

- *Job Listing Search*



Your Jobs

Software Engineer Intern
Division Technology
Role Type Internship London, United Kingdom


Your Jobs

Software Engineer Intern
Division Technology **Role Type** Internship London, United Kingdom

Business Analyst
Division Finance **Role Type** Graduate New York, United States

Cyber Security Internship
Division Technology **Role Type** Internship London, United Kingdom

- *Recruiter Job Breakdown*

- *Recruiter Job Breakdown (Mobile)*

[3.1] GDPR Compliance Report[22]

GDPR Article	Title	Considerations
Article 5	Principles relating to processing of personal data	Data subject information is processed lawfully and transparently, as outlined in our privacy policy. It is collected for legitimate purposes with explicit consent. It is adequate, relevant and limited to the purposes of processing the data subjects' applications. It is stored for a short period of time (6 months). Integrity and confidentiality are enforced to protect user data from attackers. See *Security*.
Article 6	Lawfulness of processing	Data processing is lawful since all applicants give explicit consent to have their data stored and processed for the purposes of processing an application for 6 months.
Article 9	Processing of special categories of personal data	We do not store data relating to race, ethnic origin, political opinion, religious or philosophical beliefs, trade union membership, genetics, biometrics, health, or sexual preferences.
Article 13	Information to be provided where personal data are collected from the data subject	We provide contact details for the development team inside our privacy policy. Subjects are informed that their application information will be kept for 6 months and that automated profiling (via machine learning) is used to process applications. See *Article 22*.
Article 15	Right of access by the data subject	Applicants can contact the data protection officer or data controller to see a report of all personal data related to them stored within the database. Our automated profiling system is outlined in our privacy policy.
Article 16	Right to rectification	Applicants can request rectification to their application or personal data by contacting the data controller.
Article 17	Right to erasure	Applicants can request the permanent deletion of their application or personal data by contacting the data controller. Application data stored longer than 6 months is automatically deleted when the period expires.
Article 18	Right to restriction of processing	Applicants can request a restriction of processing on their data by contacting the data controller, however, this may affect our ability to process their application.
Article 19	Notification obligation regarding rectification or erasure of personal data or restriction of processing	Applicants are informed when their application 'expires' after 6 months or when their application fails to process.
Article 20	Right to data portability	See *Article 15*. Subjects may request that their data be migrated to a third party by contacting the data controller.
Article 21	Right to object	See *Article 18*.

[22] www.gdpr-info.eu

Article 22	Automated individual decision-making, including profiling	Decisions are not made solely by automated profiling since data controller intervention is required to progress applications. Personal data outlined in Article 9 is not used to influence automated profiling. Data controllers have the ability to take manual control of automated profiling. See R11.
Article 24	Responsibility of the controller	Appropriate technical and organisational measures have been taken to protect sensitive data. See Security.
Article 25	Data protection by design and by default	Appropriate technical and organisational measures have been taken to protect sensitive data. See *Security*.
Article 26	Processor	Data processing occurs within the system. Data is only sent to a third party for storage purposes. Appropriate technical and organisational measures have been taken to protect sensitive data. See *Security*.
Article 27	Records of processing activities	The responsibilities of the data processor are maintained by the development team's data protection officer, Alistair Robinson
Article 32	Security of processing	Appropriate technical and organisational measures have been taken to protect sensitive data. See *Security*.
Article 33	Notification of a personal data breach to the supervisory authority	In the event of a data breach, a report will be submitted to the UK Information Commissioner's Office (ICO) given that our development team is based in the United Kingdom, in accordance with the UK's Data Protection Act 2018[23].
Article 34	Communication of a personal data breach to the data subject	In the event of a data breach, all users will be informed. The data protection officer will report the breach to the UK ICO[24].
Article 35	Data protection impact assessment	The development team's data protection officer, Alistair Robinson, has completed a risk assessment of the application's data protection. See *Security*.
Article 37	Designation of the data protection officer	Alistair Robinson has been assigned as the development team's data protection officer.
Article 38	Position of the data protection officer	The data protection officer has been involved in the development of the application in all sub-teams.
Article 39	Tasks of the data protection officer	The data protection officer has advised and informed other team members in decisions regarding the storage and processing of personal data throughout the development process.
Article 42	Certification	Due to the timescale of the project, it was infeasible for the system to receive official data compliance certification from a certification body.
Article 44	General principle for transfers	Personal data is not processed by third parties, only stored. MongoDB Atlas is certified as data compliant.
Article 88	Processing in the context of employment	Alistair Robinson will be responsible for ensuring compliance should the UK introduce additional legislation for data protection in the context of employment.

[23] www.gov.uk/data-protection
[24] www.ico.org.uk/

[4.1] Information Risk Assessment[25]

Vulnerability	Example Attack	Likelihood, Impact, Risk	Action taken, Residual risk (if applicable)
Unencrypted HTTP traffic	Traffic interception, Man in the Middle (MITM) attack	L: High I: Very High R: Very High	All traffic between the browser and the web server is configured to only use HTTPS, an encrypted and protected form of HTTP communication. Residual risk of impersonation due to lack of SSL certification, manageable with more development time and budget
Threat actors can fake emails from Cieve	Phishing, spear phishing attacks	L: Very High I: High R: High	We introduced a simple mutual authentication system to protect against impersonation in emails. Each user is given a plaintext word which is used in all communications with them. Since only the system has direct access to the word, this makes it harder for an attacker to impersonate the system to users. Residual risk if people disclose their phish codes.
Unsanitised user input	Cross-Site Scripting attack (XSS)	L: High I: High R: High	The Python Flask web server is configured to automatically sanitise user input. Additionally, the server's Content Security Policy (CSP) is configured to only allow JavaScript to load from trusted sources (e.g. Google's API). Risk mitigated.
Insecure Authentication	Password cracking, session hijacking attacks	L: High I: High R: High	User IDs are generated randomly, not enumerated, to mitigate session hijacking attacks. PBKDF2, a secure, industry standard hashing algorithm was used to hash salted user passwords. Complexity checks used to ensure that users only used secure passwords of a certain length and complexity. Risk mitigated.
POST requests can be falsified	Cross Site Request Forgery (CSRF)	L: Medium I: High R: High	CSRF tokens introduced. Unique tokens are generated when a page is loaded added to a user's session. When the user posts data to the server, they attach the CSRF token unique to their session. The server checks that these two tokens are equivalent and blocks any request it identifies as suspicious. Risk mitigated.
Database uses NoSQL	NoSQL Injection[26]	L: Medium I: High R: Medium	MongoDB Atlas automatically adds a layer of protection to protect against NoSQL injection attacks. Unit tests are implemented to ensure that NoSQL injection attacks fail. Risk transferred.
Third Party MongoDB Atlas database	Information disclosure attack, Denial of Service	L: Low I: High R: Low	MongoDB Atlas has undergone significant security audits to demonstrate its security[27]. The system's cluster is encrypted and can automatically create backups. Risk transferred.

[25] www.owasp.org/images/7/72/OWASP_Top_10-2017_(en).pdf.pdf
[26] www.owasp.org/index.php/Testing_for_NoSQL_injection
[27] www.mongodb.com/cloud/trust

[5.1] Team Structure

Name	Sub-team	Title	Technologies	Responsibility
Josh Hankins	Frontend	Lead Designer	HTML/CSS, JavaScript	Develop an adaptive, intelligent user interface compatible across multiple devices
Abdullah Muhammad-Kamal	Frontend	Business Analyst	HTML/CSS, JavaScript	Provide users with a seamless interface between the frontend and backend systems. Act as the primary contact for communications between the development team and the client
Nathan Hall	Backend	Project Manager	Python, Flask	Develop the web server to provide the main functionality of the product. Lead the management of the team as scrum master
Alistair Robinson	Backend	Data Protection Officer	Python, Pytest, GitHub	Maintain software tests to mitigate security risks, perform success measurement and verify data protection compliance. Oversee usage of version control and fix merge conflicts in GitHub
Felix Gaul	Data	Database Controller	Python, MongoDB Atlas	Develop a logical database schema and a database access class in the Python backend to store user data
Justas Tamulis	Data	ML Developer	Python, MongoDB Atlas	Develop an intelligent way of ranking applicants such that applicants are prioritised for relevant roles intelligently.

[5.2] Sprint Cycle Tracker

Sprint Number	Date Started	Date Completed	Frontend Target	Backend Target	Data Target
Sprint 1	08/02/19	12/02/19	Translate frontend mock up to Flask templates	Implement authentication and testing framework	Identify required database access and modification functions
Sprint 2	12/02/19	15/02/19	Translate frontend mock up to Flask templates	Start basic job creation framework, add authentication tests	Complete database access and modification functions
Sprint 3	15/02/19	19/02/19	Add basic JavaScript to templates	Fix unit testing framework	Complete database access and modification functions
Sprint 4	19/02/19	22/02/19	Add AJAX requests to completed templates	Implement job creation framework, add job creation tests	Implement machine learning feedback functions
Sprint 5	22/02/19	26/02/19	Fix page routing	Implement job search and application functionality	Finalise machine learning design
Sprint 6	26/02/19	01/03/19	Complete template designs	Complete report sections, implement machine learning into applications	Implement machine learning into applications
Sprint 7	01/03/19	04/03/19	Backend connectivity	Finalise staging process	Functions for feedback
Sprint 8	04/03/19	07/03/19	Additional features	Additional features	Functions for client dashboard