

CRISP

for

Cloud Computing (INFS 803)

by

GROUP 8

Juan Herbst (13840146)

Mike Chene (19070279)

Tommaso Cammelli (23215488)

Auckland University of Technology, Auckland, New Zealand

<i>Crisp URI</i>	<i>GitHub URI</i>
https://www.crisp.nz	https://github.com/tommecamm/aut-crisp

TABLE OF CONTENTS

Introduction	5
Project Planning	5
Project Design	6
User Experience (UX) & Screen Mock-ups	6
Cloud Service Architecture	10
Technology Stack	12
3-Tier Architecture Deployment Process & Components	13
Interactions and Sequence Diagrams	14
Schema Design	15
RESTful API Endpoints.....	21
API Schema (Swagger)	23
User Manual.....	32
Sign Up.	33
Sign In.....	34
Profile Settings	35
Change Password.....	38
Jobseeker: Job Openings	38
Jobseeker: My Job Applications	40
Recruiter: Dashboard	41
About.....	42
Summary	43
Appendix	43
Group 8 Project Contributors & Source Control	43
Total Expenditure.....	44

TABLE OF FIGURES

Figure 1 Crisp Kanban Board for Agile Project Planning.....	5
Figure 2 Sign-up Screen.....	6
Figure 3 Sign-in Screen.	7
Figure 4 Recruiter Profile Screen.....	7
Figure 5 Jobseeker Profile Screen.....	8
Figure 6 Job Posting Screen.....	9
Figure 7 Job Application Flow and Screens.....	9
Figure 8 Recruiters can find Jobseekers by Category Flow and Screens.....	10
Figure 9 Cloud Services used by Crisp.....	12
Figure 10 Continuous Integration and Deployment (CI/CD)	13
Figure 11 3-Tier Architecture Deployment Process and Components	13
Figure 12 Interactions between the User, Cognito, API Gateway, Lambda and DynamoDB.....	14
Figure 13 Interactions between the User, Cognito and S3.....	15
Figure 14 Entity Relationship Diagram (ERD).....	16
Figure 15 Swagger definitions for the Crisp API deployed in AWS API Gateway	23
Figure 16 Crisp Landing Page	32
Figure 17 Sign up as either a Job Seeker or a Recruiter.....	33
Figure 18 The verification email was sent to the sign-up email address.	34
Figure 19 Complete Sign up for an Account with Email Verification Code.....	34
Figure 20 Sign in to Crisp.....	35
Figure 21 Recruiter Profile Details	35
Figure 22 Jobseeker Profile Details	36
Figure 23 Add, Change or Delete your Profile Picture	36
Figure 24 Add, Change or Delete your Introductory Video.....	37
Figure 25 Add, Change or Remove CV	37
Figure 26 Change Password.....	38
Figure 27 Open Job Positions.	39
Figure 28 View Job Details and Apply	39
Figure 29 My Job Applications.....	40
Figure 30 Application Management Screen.....	40
Figure 31 Dashboard.....	41
Figure 32 View Job Details.....	42
Figure 33 View Applicant Details, including CV and Presentation Video, and Add Private Notes...42	
Figure 34 About Crisp.....	43
Figure 35 Total Project Expenditure	44

TABLE OF TABLES

Table 1 JSON Schema: Recruiter Profile.....	16
Table 2 JSON Schema: Jobseeker Profile.....	17
Table 3 JSON Schema: Notes.	19
Table 4 JSON Schema: Categories.	19
Table 5 JSON Schema: Jobs.	20
Table 6 REST Endpoints: Recruiter Profiles.....	21
Table 7 REST Endpoints: Jobseeker Profiles.....	22
Table 8 REST Endpoints: Notes.	22
Table 9 REST Endpoints: Categories.....	22
Table 10 REST Endpoints: Jobs.....	22
Table 11 Group 8 Contributors & Source Control Repository.....	44

Introduction

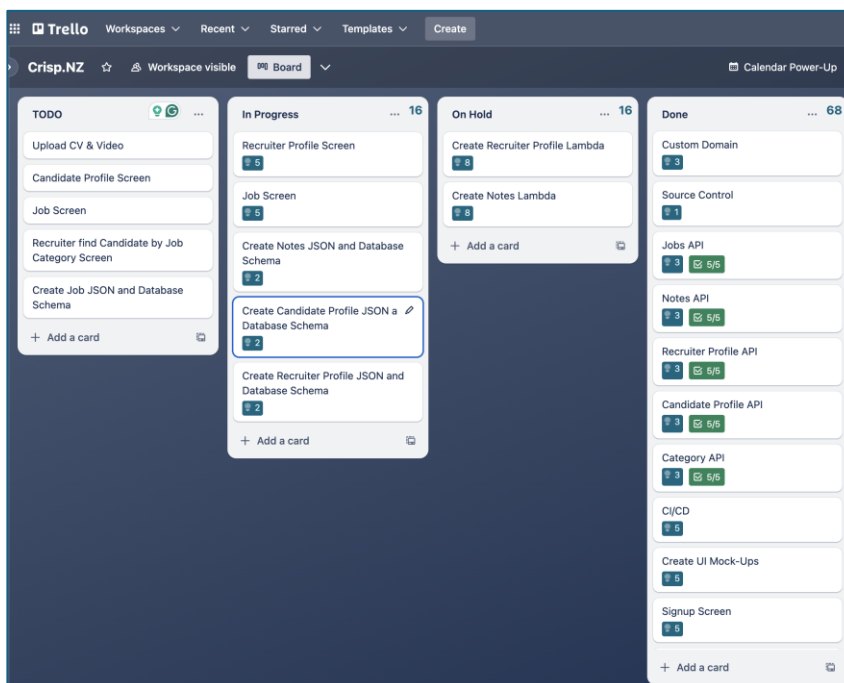
Crisp presents a complete solution aimed at enabling both jobseekers and recruiters. For jobseekers, Crisp offers an intuitive platform that facilitates the creation of personalised profiles, enhanced by the capability to upload introductory videos and traditional CVs. This multimedia integration exceeds the limitations of conventional resumes, encouraging jobseekers to highlight their competencies, professional experiences, and individuality. Conversely, on the recruiter's end, Crisp provides hiring practitioners with tools for effortless job listing posting, jobseeker evaluation, and interaction. Moreover, Crisp establishes smooth communication channels, nurturing substantive engagements between recruiters and jobseekers throughout the recruitment.

Project Planning

Scrum, a widely adopted agile framework, was used during the project design, development, and testing phases. Founded on transparency, inspection, and adaptation principles, Scrum empowers cross-functional teams to deliver high-quality products incrementally within short time frames. Scrum fosters a culture of continuous improvement and responsiveness to changing requirements through its structured ceremonies, including daily stand-ups, sprint planning, reviews, and retrospectives. Trello effectively tracked and managed the product's roadmap and maintained thorough oversight of the fortnightly sprint backlog.

Figure 1

Crisp Kanban Board for Agile Project Planning.



Project Design


User Experience (UX) & Screen Mock-ups

Screen mock-ups visually represent the user interface, giving stakeholders a tangible preview of the product's layout and functionality. They facilitate collaborative discussions, enabling teams to iterate on design concepts, gather feedback, and refine user interactions before implementation. The mock-up screens differ from the final product, reflecting the ideation phases during design and development.

The user initiates the sign-up process by providing their email address and creating a password, which they confirm to ensure accuracy. Additionally, users are prompted to select their intended role as a recruiter or job seeker during registration.

Figure 2

Sign-up Screen.



The image shows a wireframe of a web browser window. The browser has a single tab labeled "Crisp.nz". The address bar contains the URL "https://www.crisp.nz/sign-up". The main content area of the browser displays a sign-up form. The form consists of three text input fields stacked vertically, labeled "email", "Password", and "Confirm Password". Below these fields are two radio button options: "Recruiter" and "Jobseeker". At the bottom right of the form is a button labeled "Sign-up".

The user will undergo an authentication step throughout the registration process, whereby a verification code is sent to their provided email address. This verification code serves to validate the authenticity of the user's submitted email address. Upon receipt of the verification code, the user is prompted to enter it into the designated field to confirm the validity of their sign-up email address. This measure ensures the integrity and security of the sign-up process by verifying the validity of the user's contact information. Once the user is registered, they can sign in to Crisp using the email and password created during the sign-up process.

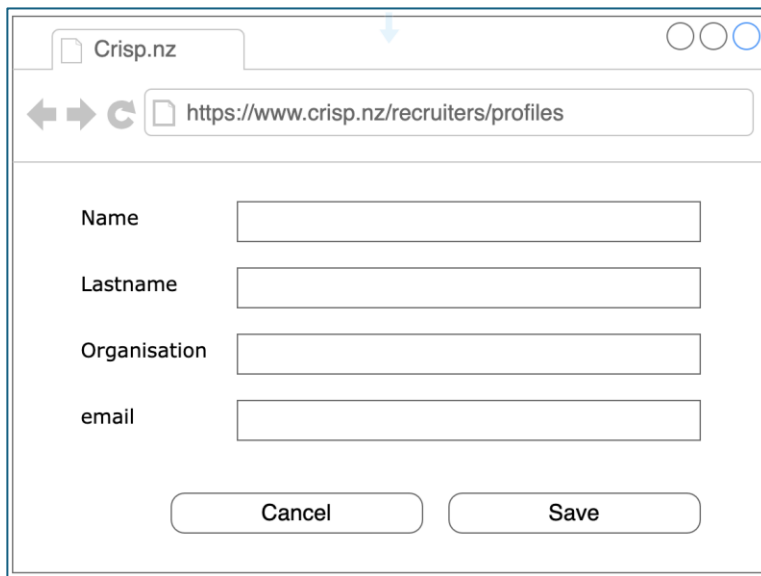
Figure 3
Sign-in Screen.



A screenshot of a web browser window showing the sign-in page for Crisp.nz. The browser's address bar displays the URL `https://www.crisp.nz/auth`. The page features two input fields: one labeled "email" and another labeled "Password". Below these fields is a "Sign-In" button. The browser window has a single tab titled "Crisp.nz" and standard window control buttons (minimize, maximize, close) in the top right corner.

If registered users sign up as recruiters, they must complete their profiles, including their name, last name, and the organisation they work for. The read-only email field will be populated with the email provided during sign-up.

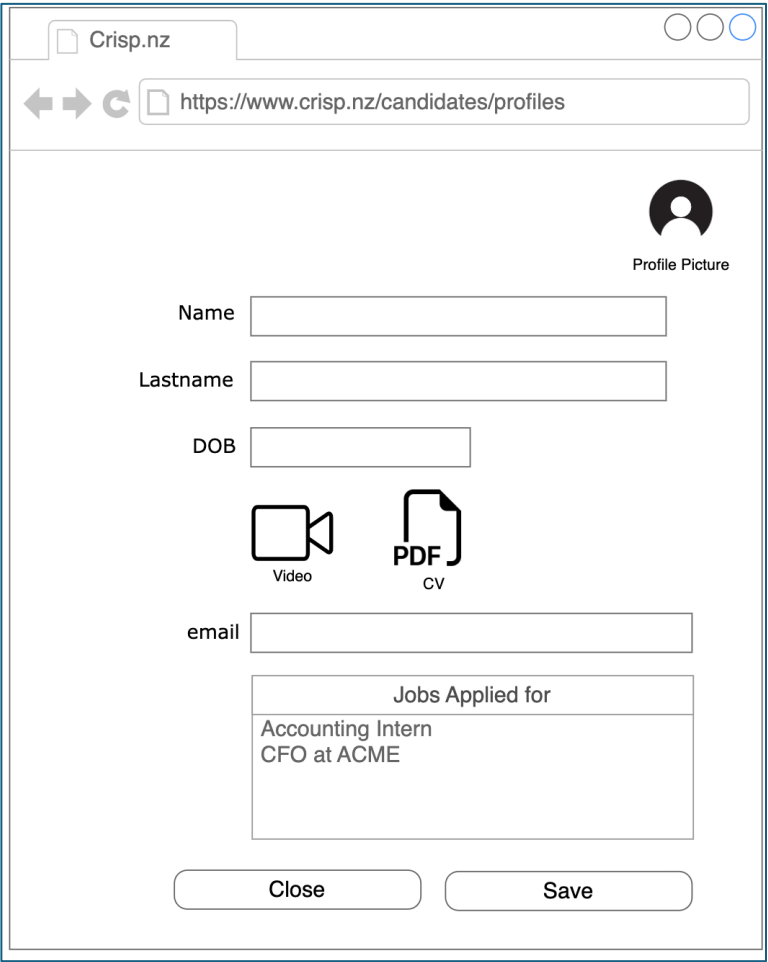
Figure 4
Recruiter Profile Screen.



A screenshot of a web browser window showing the recruiter profile page for Crisp.nz. The browser's address bar displays the URL `https://www.crisp.nz/recruiters/profiles`. The page contains four input fields: "Name", "Lastname", "Organisation", and "email". Below these fields are two buttons: "Cancel" and "Save". The browser window has a single tab titled "Crisp.nz" and standard window control buttons in the top right corner. A blue arrow points to the browser window.

Upon registering as jobseekers, users must complete their profile by providing their first name, last name, and date of birth. The read-only email field will automatically be populated with the email address submitted during registration. Users can also upload a digital or a traditional Curriculum Vitae (CV) and a profile picture. Initially, the list of job applications will be empty; however, as users apply for positions, this list will be populated accordingly.

Figure 5
Jobseeker Profile Screen.



The screenshot shows a web browser window with the address bar displaying "https://www.crisp.nz/candidates/profiles". The page contains a profile form with the following elements:

- Profile Picture:** A circular icon with a person silhouette, labeled "Profile Picture".
- Name:** A text input field.
- Lastname:** A text input field.
- DOB:** A text input field.
- Video:** A video camera icon with the label "Video" below it.
- CV:** A PDF document icon with the label "CV" below it.
- email:** A text input field.
- Jobs Applied for:** A list box containing the text "Accounting Intern" and "CFO at ACME".
- Buttons:** "Close" and "Save" buttons at the bottom.

To create a job posting, the recruiter navigates to the job posting URL, selects a job category from the pre-populated list, enters a job title, provides a job description, and specifies the offered salary.

Figure 6
Job Posting Screen.

The screenshot shows a web browser window titled 'Crisp.nz' with the address 'https://www.crisp.nz/jobs'. The page contains a form for posting a job. It has a 'Category' dropdown menu, a 'Job Title' text input, a 'Job Description' text area, and a 'Salary' field with a '\$' symbol. At the bottom of the form are two buttons: 'Cancel' and 'Save'.

The jobseeker initiates navigation to the available jobs page, where they are prompted to select a category of interest. Upon selecting a specific category, the system retrieves and displays a list of jobs corresponding to the chosen category. To view the details of a particular job, the job seeker clicks on the respective job listing, thereby accessing comprehensive job information. Within the job detail view, the job seeker can apply.

Figure 7
Job Application Flow and Screens.

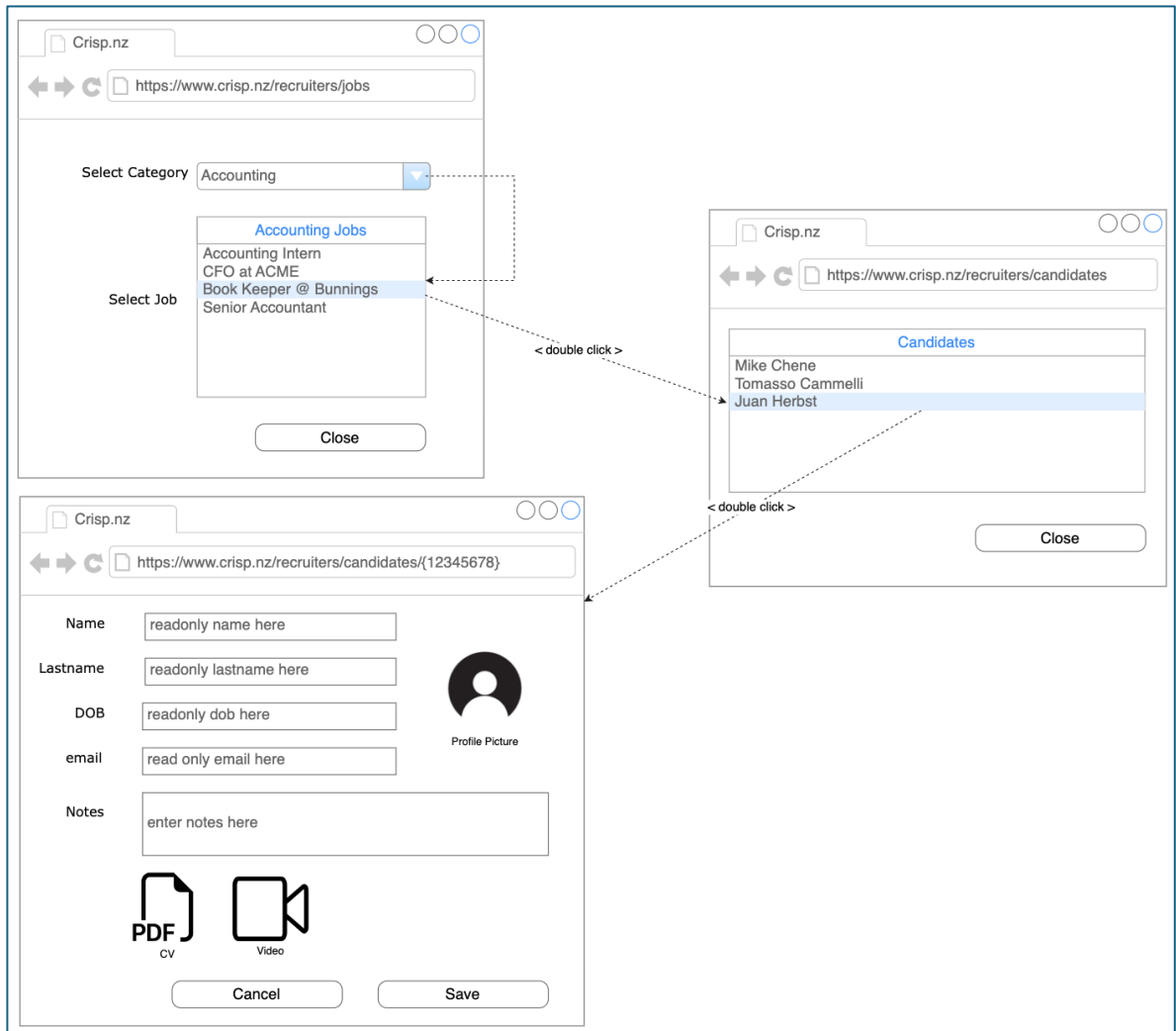
The figure illustrates the job application flow with two screenshots. The left screenshot shows the 'Select Category' dropdown set to 'Accounting', which has opened a list of 'Accounting Jobs'. The jobs listed are 'Accounting Intern', 'CFO at ACME', 'Book Keeper @ Bunnings', and 'Senior Accountant'. A dashed arrow points from 'Book Keeper @ Bunnings' to the right screenshot. The right screenshot shows the job details for 'Book Keeper @ Bunnings'. It includes fields for 'Job Title' (filled with 'Book Keeper @ Bunnings'), 'Job Description' (readonly), and 'Salary' (readonly). At the bottom are 'Cancel' and 'Apply' buttons. A '< double click >' label with an arrow points to the 'Book Keeper @ Bunnings' job listing in the left window, indicating the action that triggers the right window.

The recruiter navigates to the job category screen to find jobseekers who have applied for a specific role. Here, they select a job category from a pre-populated drop-down list, which will subsequently populate a list of jobs within that category. A pop-up window will appear when double-clicking on a

job, displaying a list of candidates who have applied for that position. Double-clicking on a particular candidate will open their profile, showing their first name, last name, date of birth, and email (the latter being read-only). The only editable field is a text box where the recruiter can enter notes regarding the job seeker. Additionally, the recruiter can download both the candidate's digital and traditional CVs.

Figure 8

Recruiters can find Jobseekers by Category Flow and Screens.



Cloud Service Architecture

This paper introduces Amazon Web Services (AWS) as the cloud platform of choice for the Crisp project. Leveraging AWS's robust, cost-effective infrastructure, extensive service offerings, and proven reliability, the CRISP project aims to enhance data privacy, ensure high availability, and facilitate seamless scalability. The following sections will explore the specific AWS services utilised

in the Crisp project, demonstrating how each service contributes to the project's overall objectives and technical requirements.

Cognito is an identity management service offered by AWS. It facilitates secure user authentication and authorisation for web and mobile applications. Amazon Cognito provides users various features, including user sign-up, sign-in, account recovery, profile management, and data synchronisation across multiple devices. Furthermore, it supports multi-factor authentication (MFA), social identity provider integration, and fine-grained access control policies.

Route 53 is a highly scalable and reliable Domain Name System (DNS) provided by AWS. Route 53 effectively translates human-readable domain names into corresponding numeric IP addresses by routing user requests across the Internet. Route 53 offers a range of functions, including domain registration, DNS health monitoring, traffic routing policies, and domain name system security extensions (DNSSEC).

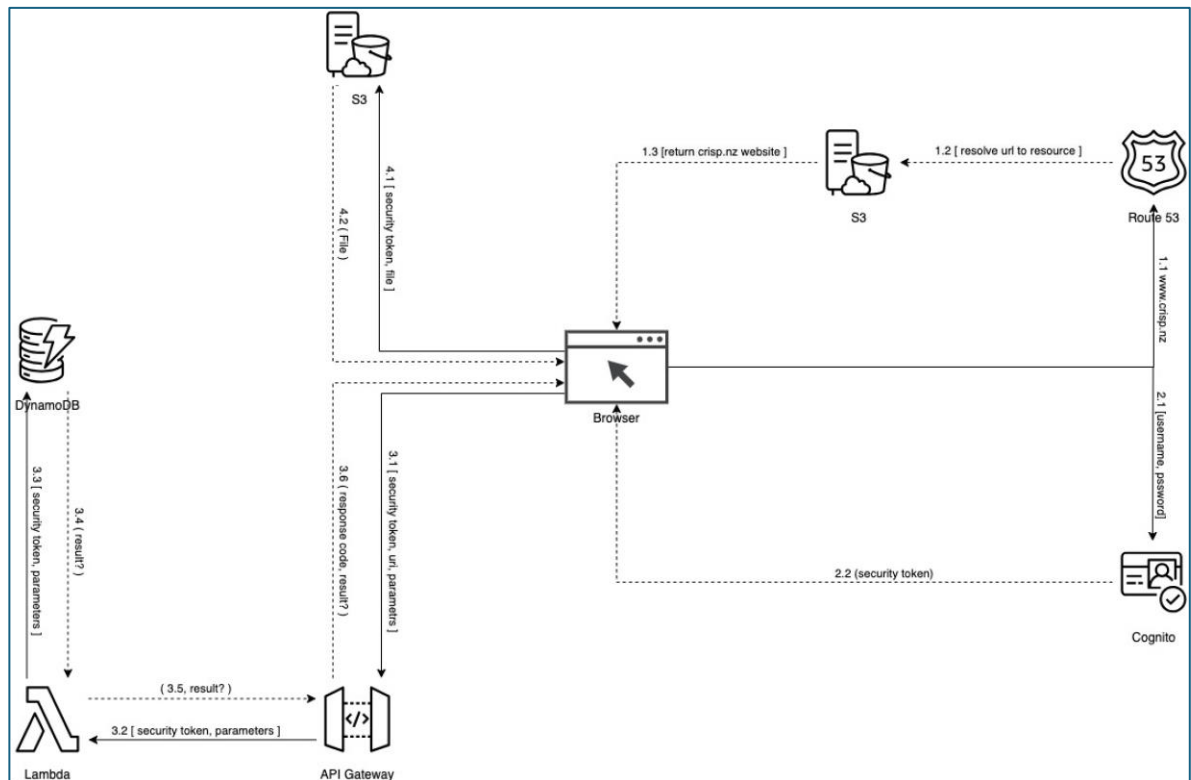
Simple Storage Service (S3) provides a secure and reliable platform for storing and retrieving objects such as large files. With its distributed architecture spanning multiple geographic regions, Amazon S3 ensures high durability and availability, making it suitable for various applications, from data backup and archival to content delivery and big data analytics.

API Gateway is a fully managed service provided by AWS that allows developers to create, publish, maintain, monitor, and secure APIs at any scale. As a front door for applications to access data, business logic, or functionality from backend services, API Gateway simplifies the process of building and managing APIs. Leveraging API Gateway, developers can define RESTful APIs or WebSocket APIs, configure endpoints, handle authentication and authorisation, and enforce usage policies.

Lambda is a serverless computing service that enables developers to run code without provisioning or managing servers. Lambda automatically scales and manages the compute resources needed to execute the code in response to incoming requests. Lambda supports various programming languages, including Node.js, Python, Java, and Go, allowing developers to write functions that respond to events triggered by AWS services or custom events.

DynamoDB is a fully managed NoSQL database service designed to provide high performance, scalability, and reliability for applications requiring low-latency data access.

Figure 9
Cloud Services used by Crisp.



Technology Stack

The Lambda functions were developed using Node.js version 18 and Express, a lightweight web framework characterised by its rapid processing capabilities and minimalistic design philosophy. Furthermore, TypeScript and React were selected to facilitate the development of dynamic user interfaces for front-end applications. Vite was used to optimise the build process, ensuring efficient compilation of project assets. In addition, Tailwind CSS was adopted to streamline the styling of the application, promoting swift responsiveness and usability. Testing frameworks such as Vitest for unit testing and Playwright for end-to-end testing were integrated into the development workflow to maintain code integrity.

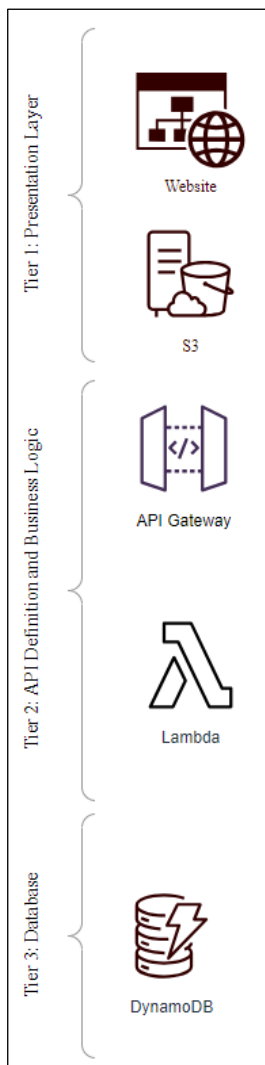
GitHub served as the designated platform for source control management, facilitating versioning and collaborative efforts among team members. Continuous Integration and Deployment Pipelines (CI/CD) were established using Amplify to streamline the deployment of the back and front-end components triggered by successful pull requests and merges to the main branch, maintaining a consistent production update cycle.

Figure 10
Continuous Integration and Deployment (CI/CD)

Deployment history		Deployed backend resources	
Deployment history (4)		Search...	
Name	Status	Commit message	Started at
Deployment 4	Deployed	Merge pull request #3 from tom...	5/9/2024, 4:17 PM
Deployment 3	Failed	Merge pull request #2 from tom...	5/9/2024, 3:57 PM
Deployment 2	Deployed	Merge pull request #1 from tom...	3/27/2024, 5:20 PM
Deployment 1	Deployed	This is an autogenerated message	3/20/2024, 6:41 PM
		Rows per page 15	

3-Tier Architecture Deployment Process & Components

Figure 11 3-Tier Architecture Deployment Process and Components



AWS Amplify PaaS software development framework greatly simplifies the deployment of the Crisp application's three-tier architecture. This architecture comprises three distinct tiers. The first is the presentation tier, where the static website is hosted on Amazon S3. The second tier is the application logic tier, involving the definition and management of APIs through Amazon API Gateway and the execution of backend processes via AWS Lambda functions. The final tier is the data tier, which relies on Amazon DynamoDB for data persistence.

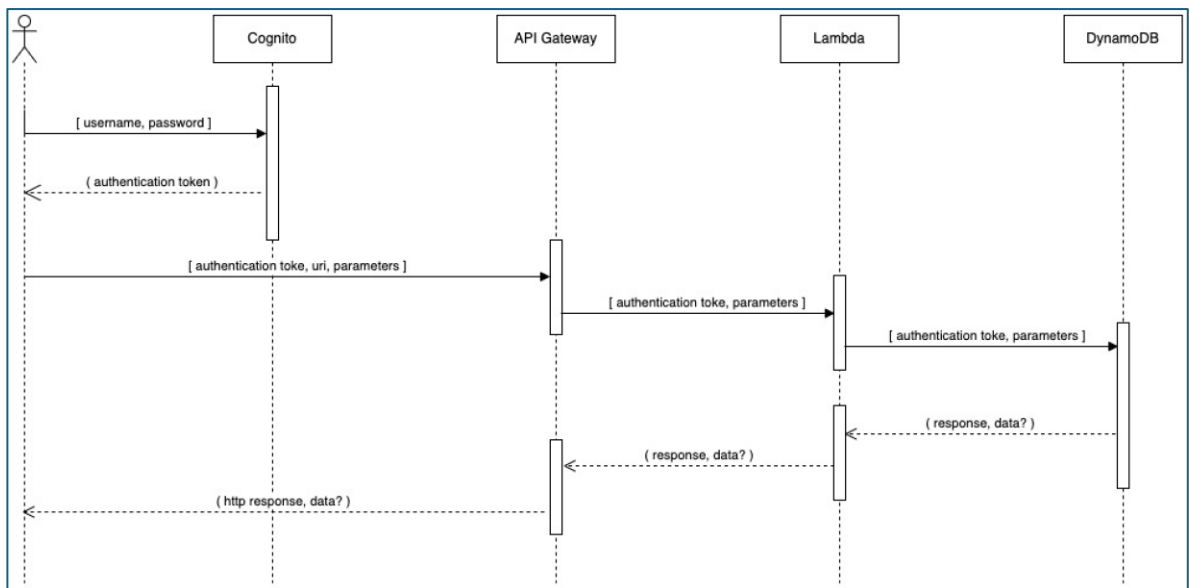
Amplify uses AWS CloudFormation's Infrastructure as Code (IaC) methods to automate the deployment and configuration of these components, ensuring a streamlined and consistent setup across all tiers. Additionally, AWS Amplify integrates with GitHub to manage deployment triggers via webhooks. When changes are merged into the main branch of the Crisp GitHub repository, these webhooks notify Amplify to initiate the Continuous Integration and Continuous Deployment (CI/CD) pipeline. This automated pipeline process involves fetching the latest code, building the application, running tests, and deploying updates. This integration ensures that the Crisp application remains consistently updated and operates reliably while minimising the need for manual intervention in the deployment process.

Interactions and Sequence Diagrams

The user initiates the authentication process by validating their email address and password against Cognito. Upon successful authentication, a security token is issued to the user, allowing them to perform authenticated actions within AWS. Requests are made by providing the security token and URL to the desired REST resource, facilitated by API Gateway. This service routes requests to a serverless backend implemented using Lambda, where business logic is executed. Lambda functions interact with DynamoDB to retrieve and process data. Retrieved data is transformed and subjected to further business logic within Lambda functions before being delivered back to the user's browser through AWS API Gateway, completing the transaction cycle.

Figure 12

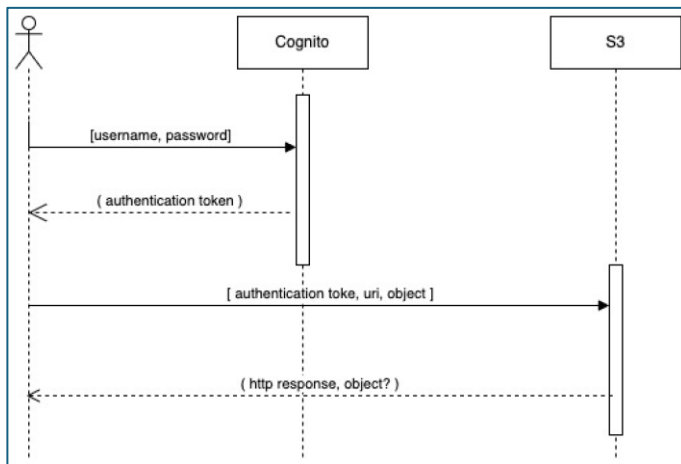
Interactions between the User, Cognito, API Gateway, Lambda and DynamoDB.



The authentication process begins as the user submits their email address and password for validation against Cognito. Upon validation, an authentication token is provided to the user, granting access to authenticated actions within AWS. Objects like files, videos, and images are also uploaded to S3. Conversely, files, videos, and images are retrieved from the S3 storage repository through a download process.

Figure 13

Interactions between the User, Cognito and S3.

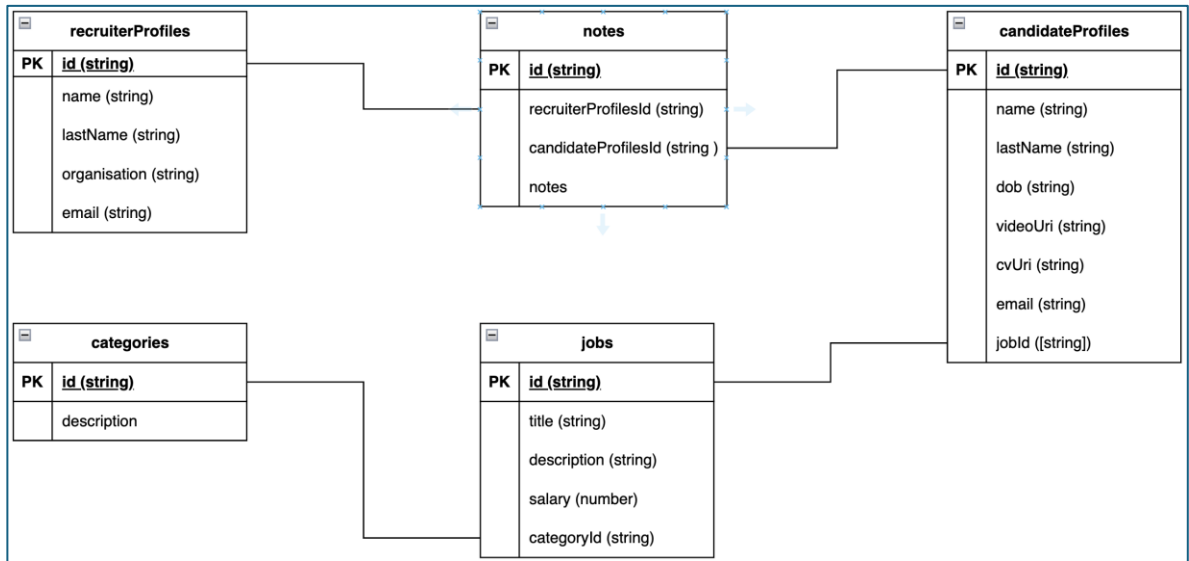


Schema Design

Schema design is an essential component of database management. It encompasses the arrangement and organisation of data within a database system. It involves creating data models that delineate the structure, relationships, and constraints of the data stored in the database.

In the context of NoSQL databases like DynamoDB, it's noteworthy that conventional relationships between tables, as seen in relational databases, are absent. Instead, NoSQL databases often employ informal relationships or data denormalisation to optimise data access patterns and enhance performance.

Figure 14
Entity Relationship Diagram (ERD)



The JSON schema for a recruiter profile delineates the structure and organisation of personal data stored for each recruiter within the system. This schema includes several vital functions. The "id" field, represented as a string, serves as a unique identifier for each recruiter profile, enabling efficient retrieval and management of individual records. Additionally, the schema incorporates fields such as "name" and "lastName," both string types, to capture the recruiter's first and last names, respectively. The "organisation" field, also of string type, facilitates the storage of information regarding the recruiter's affiliated organisation or company. Furthermore, the schema includes an "email" field, represented as a string, designed to store the recruiter's email address, providing a means for communication and correspondence.

Table 1
JSON Schema: Recruiter Profile.

```

1. {
2.   "$schema": "http://json-schema.org/draft-04/schema#",
3.   "type": "object",
4.   "properties": {
5.     "id": {
6.       "type": "string."
7.     },
8.     "name": {
9.       "type": "string"
10.    },

```



```

11.     "lastName": {
12. "type": "string"
13. },
14. "organisation": {
15. "type": "string"
16. },
17. "email": {
18. "type": "string"
19. }
20. },
21. "required": [
22. "id",
23. "name",
24. "lastName",
25. "organisation",
26. "email"
27. ]
28. }

```

The JSON schema for a jobseeker profile encompasses essential fields intended to capture relevant personal data for jobseekers within the Crisp project. These fields include "id" as a string, serving as a unique identifier for each jobseeker profile; "name" and "lastName" as strings to record the jobseeker's given and family names, respectively; and "dob" as a string to denote the jobseeker's date of birth. Additionally, the schema features "email" as a string, facilitating communication with jobseekers via their designated email addresses. Notably, the schema incorporates "videoUri" and "cvUri" fields to store Uniform Resource Identifiers (URIs) pointing to videos and CVs stored in S3, respectively. Furthermore, the schema accommodates an array of "jobIds" to record the identifiers of jobs to which jobseekers have applied, facilitating tracking and managing job seeker applications.

Table 2

JSON Schema: Jobseeker Profile.

```

1. {
2.   "$schema": "http://json-schema.org/draft-04/schema#",
3.   "type": "object",
4.   "properties": {
5.     "id": {
6.       "type": "string"
7.     },
8.     "name": {

```

```

9.   "type": "string"
10. },
11.   "lastName": {
12. "type": "string"
13. },
14. "email": {
15. "type": "string"
16. },
17. "dob": {
18. "type": "string"
19. },
20. "videoUri": {
21. "type": "string"
22. },
23. "cvUri": {
24. "type": "string"
25. },
26. "jobId": {
27. "type": "array",
28. "items": [
29.   {
30. "type": "string"
31.   }
32. ]
33. }
34. },
35. "required": [
36. "id",
37. "name",
38. "lastName",
39. "email",
40. "dob",
41. "videoUri",
42. "cvUri",
43. "jobId"
44. ]
45. }

```

The JSON schema for a note encapsulates key fields necessary for recording and managing notes associated with interactions between recruiting jobseekers. This schema includes "id" as a string, a unique identifier for each note entry. Additionally, the schema features "recruiterProfileId" and "jobseekerProfileId" as strings, facilitating the association of notes with the specific recruiter and jobseeker profiles, respectively.

Table 3

JSON Schema: Notes.

```
1. {
2.   "$schema": "http://json-schema.org/draft-04/schema#",
3.   "type": "object",
4.   "properties": {
5.     "id": {
6.       "type": "string"
7.     },
8.     "recruiterProfileId": {
9.       "type": "string"
10.    },
11.    "jobseekerProfileId": {
12.      "type": "string"
13.    },
14.    "note": {
15.      "type": "string"
16.    }
17.  },
18.  "required": [
19.    "id",
20.    "recruiterProfileId",
21.    "jobseekerProfileId",
22.    "note"
23.  ]
24. }
```

The JSON schema for a category is a foundational structure for organising and managing job categories. This schema encompasses essential fields for effectively categorising and describing different job types. The schema includes “id” as a string and functions as a unique identifier for each category entry, ensuring distinct identification and retrieval of specific categories. Additionally, the “description” field, a string, provides a textual representation of the category.

Table 4

JSON Schema: Categories.

```
1. {
2.   "$schema": "http://json-schema.org/draft-04/schema#",
3.   "type": "object",
4.   "properties": {
5.     "id": {
6.       "type": "string"
```

```

7.     },
8.   "description": {
9.     "type": "string"
10.   }
11. },
12. "required": [
13.   "id",
14.   "description"
15. ]
16. }

```

The JSON schema for representing job entities involves several fields for job descriptions. Firstly, the “id” attribute, a string, is a unique identifier for each job entry, ensuring its distinct identification and retrieval within the system. Subsequently, the “title” field, also specified as a string, encapsulates a prescriptive title for the job role. Moreover, the “description” attribute, a string, provides a detailed overview of the job’s responsibilities, qualifications, and other pertinent information. Additionally, the “salary” field, represented as a number, denotes the monetary compensation associated with the job. Lastly, the “categoryId” attribute, designated as a string, associates the job with a specific category, enabling efficient categorisation and organisation of job listings based on their respective domains or industries.

Table 5

JSON Schema: Jobs.

```

1. {
2.   "$schema": "http://json-schema.org/draft-04/schema#",
3.   "type": "object",
4.   "properties": {
5.     "id": {
6.       "type": "string"
7.     },
8.     "title": {
9.       "type": "string"
10.    },
11.    "description": {
12.      "type": "string"
13.    },
14.    "salary": {
15.      "type": "string"
16.    },
17.    "categoryId": {
18.      "type": "string"

```

```

19.     }
20.   },
21.   "required": [
22.     "id",
23.     "title",
24.     "description",
25.     "salary",
26.     "categoryId"
27.   ]
28. }

```

RESTful API Endpoints.

The Representational State Transfer (REST) Application Programming Interfaces (APIs) are deployed utilising the Hypertext Transfer Protocol Secure (HTTPS), ensuring secure communication and safeguarding the integrity and confidentiality of data transferred across the network. This implementation of HTTPS encryption helps prevent unauthorised access and interception of sensitive information, thereby enhancing the overall security posture of the system. Within the context of the Crisp project, the <https://api.crisp.nz> custom domain serves as the deployment endpoint for the REST APIs, facilitating interactions with backend services. In contrast, the application's front-end components are deployed on the <https://crisp.nz> domain, which hosts the user interface and facilitates user interactions with the web application's features and content.

Table 6

REST Endpoints: Recruiter Profiles.

Endpoint	Verb	Action
https://api.crisp.co.nz/recruiters/profiles	POST	Create a Profile.
https://api.crisp.co.nz/recruiters/profiles	GET	Get all Profiles.
https://api.crisp.co.nz/recruiters/profiles/{profileId}	GET	Get a Profile by ID.
https://api.crisp.co.nz/recruiters/profiles/	PUT	Update a Profile.
https://api.crisp.co.nz/recruiters/profiles/{profileId}	DELETE	Delete a Profile by ID.

Table 7

REST Endpoints: Jobseeker Profiles.

Endpoint	Verb	Action
https://api.crisp.co.nz/jobseekers/profiles	POST	Create a Profile.
https://api.crisp.co.nz/jobseekers/profiles	GET	Get all Profiles.
https://api.crisp.co.nz/jobseekers/profiles/{profileId}	GET	Get a Profile by ID.
https://api.crisp.co.nz/jobseekers/profiles/	PUT	Update a Profile.
https://api.crisp.co.nz/jobseekers/profiles/{profileId}	DELETE	Delete a Profile by ID.

Table 8

REST Endpoints: Notes.

Endpoint	Verb	Action
https://api.crisp.co.nz/notes/	POST	Create a Note.
https://api.crisp.co.nz/notes/	GET	Get all Notes.
https://api.crisp.co.nz/notes/{noteId}	GET	Get a Note by ID.
https://api.crisp.co.nz/notes/	PUT	Update a Note.
https://api.crisp.co.nz/notes/{noteId}	DELETE	Delete a Note by ID.

Table 9

REST Endpoints: Categories.

Endpoint	Verb	Action
https://api.crisp.co.nz/categories/	POST	Create a Category.
https://api.crisp.co.nz/categories/	GET	Get all Categories.
https://api.crisp.co.nz/categories/{categoryId}	GET	Get a Category by ID.
https://api.crisp.co.nz/categories/	PUT	Update a Category.
https://api.crisp.co.nz/categories/{categoryId}	DELETE	Delete a Category by ID.

Table 10

REST Endpoints: Jobs.

Endpoint	Verb	Action
https://api.crisp.co.nz/jobs/	POST	Create a Job.
https://api.crisp.co.nz/jobs/	GET	Get all Jobs.

https://api.crisp.co.nz/jobs/{jobId}	GET	Get a Job by ID.
https://api.crisp.co.nz/jobs/	PUT	Update a Job.
https://api.crisp.co.nz/jobs/{jobId}	DELETE	Delete a Job by ID.

API Schema (Swagger)

The SWAGGER schema for Crisp provides a comprehensive, machine-readable description of the Crisp API, enabling humans and computers to understand a web service's capabilities and structure without direct access to the source code.

Figure 15

Swagger definitions for the Crisp API deployed in AWS API Gateway

```

---
swagger: "2.0"
info:
  version: "2018-05-24T17:52:00Z"
  title: "crispApi"
host: "api.crisp.nz"
schemes:
- "https"
paths:
  /candidates/profiles:
    options:
      consumes:
      - "application/json"
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Headers:
              type: "string"
      x-amazon-apigateway-any-method:
        consumes:
        - "application/json"
        produces:
        - "application/json"
        parameters:
        - in: "body"
          name: "RequestSchema"

```

```

    required: true
    schema:
      $ref: "#/definitions/RequestSchema"
  responses:
    "200":
      description: "200 response"
      schema:
        $ref: "#/definitions/ResponseSchema"
  security:
    - sigv4: []
/candidates/profiles/{proxy+}:
  options:
    consumes:
      - "application/json"
    parameters:
      - name: "proxy"
        in: "path"
        required: true
        type: "string"
    responses:
      "200":
        description: "200 response"
        headers:
          Access-Control-Allow-Origin:
            type: "string"
          Access-Control-Allow-Methods:
            type: "string"
          Access-Control-Allow-Headers:
            type: "string"
  x-amazon-apigateway-any-method:
    consumes:
      - "application/json"
    produces:
      - "application/json"
    parameters:
      - name: "proxy"
        in: "path"
        required: true
        type: "string"
      - in: "body"
        name: "RequestSchema"
        required: true
        schema:
          $ref: "#/definitions/RequestSchema"
    responses:
      "200":
        description: "200 response"

```



```

        schema:
          $ref: "#/definitions/ResponseSchema"
      security:
        - sigv4: []
    /categories:
      options:
        consumes:
          - "application/json"
        responses:
          "200":
            description: "200 response"
            headers:
              Access-Control-Allow-Origin:
                type: "string"
              Access-Control-Allow-Methods:
                type: "string"
              Access-Control-Allow-Headers:
                type: "string"
      x-amazon-apigateway-any-method:
        consumes:
          - "application/json"
        produces:
          - "application/json"
        parameters:
          - in: "body"
            name: "RequestSchema"
            required: true
            schema:
              $ref: "#/definitions/RequestSchema"
        responses:
          "200":
            description: "200 response"
            schema:
              $ref: "#/definitions/ResponseSchema"
        security:
          - sigv4: []
    /categories/{proxy+}:
      options:
        consumes:
          - "application/json"
        parameters:
          - name: "proxy"
            in: "path"
            required: true
            type: "string"
        responses:
          "200":

```

```

        description: "200 response"
        headers:
            Access-Control-Allow-Origin:
                type: "string"
            Access-Control-Allow-Methods:
                type: "string"
            Access-Control-Allow-Headers:
                type: "string"
x-amazon-apigateway-any-method:
    consumes:
        - "application/json"
    produces:
        - "application/json"
    parameters:
        - name: "proxy"
          in: "path"
          required: true
          type: "string"
        - in: "body"
          name: "RequestSchema"
          required: true
          schema:
            $ref: "#/definitions/RequestSchema"
    responses:
        "200":
            description: "200 response"
            schema:
                $ref: "#/definitions/ResponseSchema"
    security:
        - sigv4: []
/jobs:
    options:
        consumes:
            - "application/json"
        responses:
            "200":
                description: "200 response"
                headers:
                    Access-Control-Allow-Origin:
                        type: "string"
                    Access-Control-Allow-Methods:
                        type: "string"
                    Access-Control-Allow-Headers:
                        type: "string"
x-amazon-apigateway-any-method:
    consumes:
        - "application/json"

```

```

    produces:
      - "application/json"
    parameters:
      - in: "body"
        name: "RequestSchema"
        required: true
        schema:
          $ref: "#/definitions/RequestSchema"
    responses:
      "200":
        description: "200 response"
        schema:
          $ref: "#/definitions/ResponseSchema"
    security:
      - sigv4: []
  /jobs/{proxy+}:
    options:
      consumes:
        - "application/json"
      parameters:
        - name: "proxy"
          in: "path"
          required: true
          type: "string"
      responses:
        "200":
          description: "200 response"
          headers:
            Access-Control-Allow-Origin:
              type: "string"
            Access-Control-Allow-Methods:
              type: "string"
            Access-Control-Allow-Headers:
              type: "string"
    x-amazon-apigateway-any-method:
      consumes:
        - "application/json"
      produces:
        - "application/json"
      parameters:
        - name: "proxy"
          in: "path"
          required: true
          type: "string"
        - in: "body"
          name: "RequestSchema"
          required: true

```

```

    schema:
      $ref: "#/definitions/RequestSchema"
  responses:
    "200":
      description: "200 response"
      schema:
        $ref: "#/definitions/ResponseSchema"
  security:
    - sigv4: []
/notes:
  options:
    consumes:
      - "application/json"
  responses:
    "200":
      description: "200 response"
      headers:
        Access-Control-Allow-Origin:
          type: "string"
        Access-Control-Allow-Methods:
          type: "string"
        Access-Control-Allow-Headers:
          type: "string"
x-amazon-apigateway-any-method:
  consumes:
    - "application/json"
  produces:
    - "application/json"
  parameters:
    - in: "body"
      name: "RequestSchema"
      required: true
      schema:
        $ref: "#/definitions/RequestSchema"
  responses:
    "200":
      description: "200 response"
      schema:
        $ref: "#/definitions/ResponseSchema"
  security:
    - sigv4: []
/notes/{proxy+}:
  options:
    consumes:
      - "application/json"
  parameters:
    - name: "proxy"

```

```

    in: "path"
    required: true
    type: "string"
  responses:
    "200":
      description: "200 response"
      headers:
        Access-Control-Allow-Origin:
          type: "string"
        Access-Control-Allow-Methods:
          type: "string"
        Access-Control-Allow-Headers:
          type: "string"
  x-amazon-apigateway-any-method:
    consumes:
      - "application/json"
    produces:
      - "application/json"
    parameters:
      - name: "proxy"
        in: "path"
        required: true
        type: "string"
      - in: "body"
        name: "RequestSchema"
        required: true
        schema:
          $ref: "#/definitions/RequestSchema"
    responses:
      "200":
        description: "200 response"
        schema:
          $ref: "#/definitions/ResponseSchema"
    security:
      - sigv4: []
/recruiters/profiles:
  options:
    consumes:
      - "application/json"
    responses:
      "200":
        description: "200 response"
        headers:
          Access-Control-Allow-Origin:
            type: "string"
          Access-Control-Allow-Methods:
            type: "string"

```

```

        Access-Control-Allow-Headers:
            type: "string"
x-amazon-apigateway-any-method:
    consumes:
        - "application/json"
    produces:
        - "application/json"
    parameters:
        - in: "body"
          name: "RequestSchema"
          required: true
          schema:
              $ref: "#/definitions/RequestSchema"
    responses:
        "200":
            description: "200 response"
            schema:
                $ref: "#/definitions/ResponseSchema"
    security:
        - sigv4: []
/recruiters/profiles/{proxy+}:
    options:
        consumes:
            - "application/json"
        parameters:
            - name: "proxy"
              in: "path"
              required: true
              type: "string"
        responses:
            "200":
                description: "200 response"
                headers:
                    Access-Control-Allow-Origin:
                        type: "string"
                    Access-Control-Allow-Methods:
                        type: "string"
                    Access-Control-Allow-Headers:
                        type: "string"
x-amazon-apigateway-any-method:
    consumes:
        - "application/json"
    produces:
        - "application/json"
    parameters:
        - name: "proxy"
          in: "path"

```

```

        required: true
        type: "string"
      - in: "body"
        name: "RequestSchema"
        required: true
        schema:
          $ref: "#/definitions/RequestSchema"
    responses:
      "200":
        description: "200 response"
        schema:
          $ref: "#/definitions/ResponseSchema"
    security:
      - sigv4: []
  securityDefinitions:
    sigv4:
      type: "apiKey"
      name: "Authorization"
      in: "header"
      x-amazon-apigateway-authtype: "awsSigv4"
  definitions:
    RequestSchema:
      type: "object"
      required:
        - "request"
      properties:
        request:
          type: "string"
      title: "Request Schema"
    ResponseSchema:
      type: "object"
      required:
        - "response"
      properties:
        response:
          type: "string"
      title: "Response Schema"

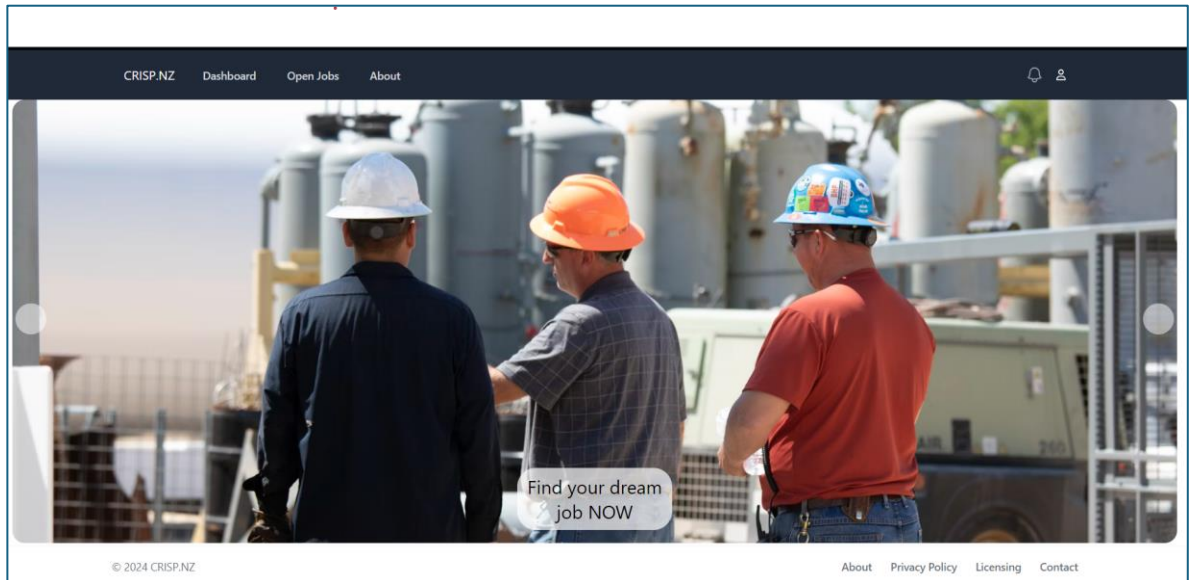
```

User Manual

Our platform is dedicated to professionals across various industries, helping them find their dream jobs efficiently and effectively. Whether you're an employer looking for skilled workers or a job seeker aiming to land your ideal role, Crisp is here to support you every step.

Figure 16

Crisp Landing Page



Sign Up.

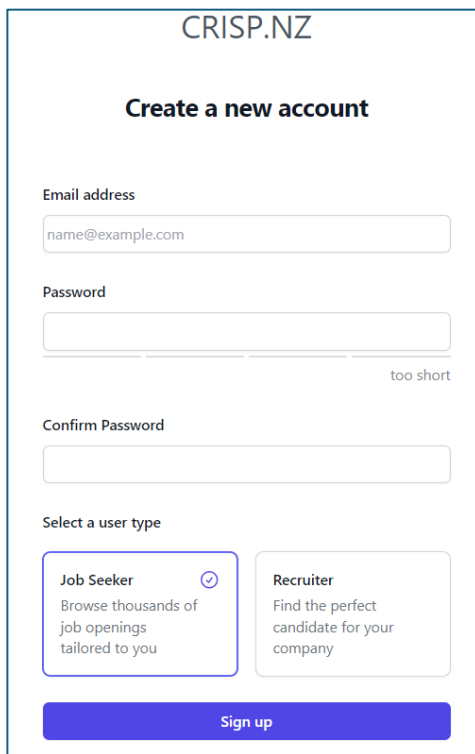
This screen allows you to create a new account on Crisp as either a Recruiter or a Job Seeker.

Follow the steps below to set up your account:

First, enter your valid email address, which will be used for account verification and communication. Next, create a strong password for your account. To ensure security, the password should be at least eight characters long. Enter your chosen password, where a visual on-screen indicator will show how strong the password is. Then, re-enter the same password to verify it. Choose whether you are a "Job Seeker" or a "Recruiter" by selecting the appropriate option. Jobseekers can browse job openings tailored to their skills and preferences, whereas Recruiters can access our extensive talent pool to find the perfect candidate for your company.

Figure 17

Sign up as either a Job Seeker or a Recruiter.



The screenshot shows the 'Create a new account' form on the Crisp.NZ website. The form is titled 'CRISP.NZ' at the top. Below the title is the heading 'Create a new account'. The form contains the following fields and options:

- Email address:** A text input field with the placeholder 'name@example.com'.
- Password:** A text input field with a strength indicator below it showing 'too short'.
- Confirm Password:** A text input field.
- Select a user type:** Two radio button options:
 - Job Seeker:** Selected by default (indicated by a blue checkmark). Description: 'Browse thousands of job openings tailored to you'.
 - Recruiter:** Description: 'Find the perfect candidate for your company'.
- Sign up:** A blue button at the bottom of the form.

Once you have filled in all the required fields and selected your user type, click the "Sign up" button to create your account. After clicking "Sign up," you will receive a confirmation email with instructions on verifying your account. Follow the instructions in the email to complete your registration.

Figure 18

The verification email was sent to the sign-up email address.

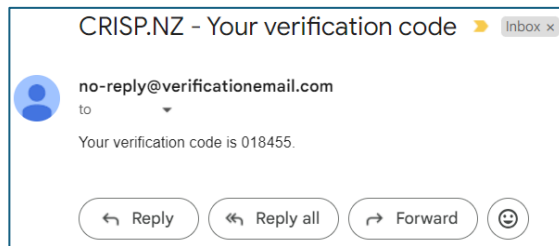


Figure 19

Complete Sign up for an Account with Email Verification Code.

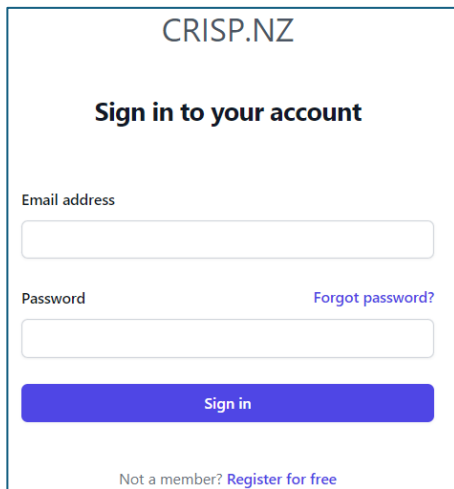
A screenshot of a web form titled "CRISP.NZ" and "Create a new account". Below the title is the heading "Confirm your account". The text says "A confirmation code was sent to your inbox, please write the code in the form below." There are two input fields: "Email address" with the value "ij +3@gmail.com" and "Confirmation code" with the value "235874". At the bottom is a blue button labeled "Confirm account".

Sign In

The Sign-In screen allows you to access your Crisp account. Follow the steps below to log in: First, enter your registered email address. Next, type your account password. After entering your email address and password, click the "Sign in" button to access your account. If you are not a member, click the "Register for free" link to create a new account.

Figure 20

Sign in to Crisp.



The image shows a sign-in form for CRISP.NZ. At the top, the text "CRISP.NZ" is displayed. Below it, the heading "Sign in to your account" is centered. The form contains two input fields: "Email address" and "Password". To the right of the password field is a link that says "Forgot password?". Below the input fields is a blue button labeled "Sign in". At the bottom of the form, there is a link that says "Not a member? Register for free".

Profile Settings

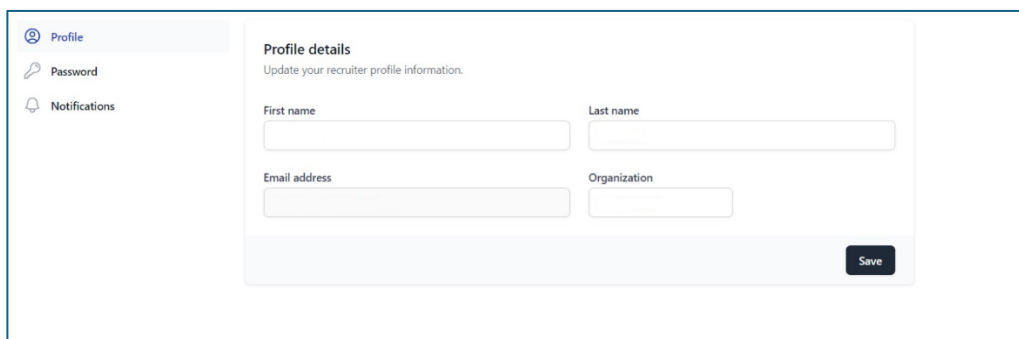
Click on the "Profile" option in the menu to navigate to the Profile Settings page. This page allows you to add or update your personal information as a Recruiter or Jobseeker.

Recruiter: Profile Details

The section allows users to update their personal and organisational information. The profile details interface includes fields for the first name, last name, email address, and organisation name. The email address is read-only as it is captured during the signup process. Users enter their information in the respective fields to update the profile and click the "Save" button.

Figure 21

Recruiter Profile Details



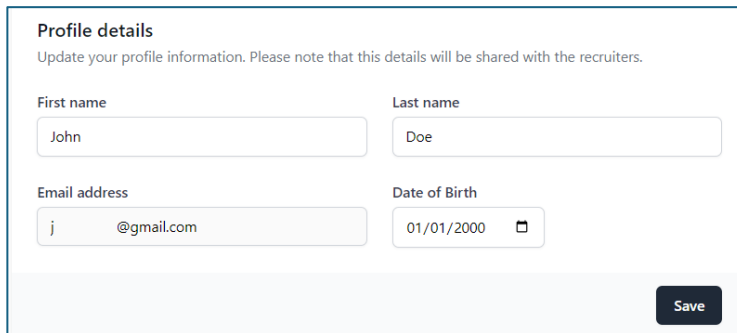
The image shows the "Recruiter Profile Details" form. On the left, there is a sidebar menu with three options: "Profile" (selected), "Password", and "Notifications". The main content area is titled "Profile details" and has a subtitle "Update your recruiter profile information.". Below this, there are four input fields arranged in a 2x2 grid: "First name", "Last name", "Email address", and "Organization". The "Email address" field is read-only. At the bottom right of the form, there is a "Save" button.

Jobseeker: Profile Details

Enter your first name, followed by your last name. The Email address is read-only, as it is captured during the signup process. Finally, use the date picker in the "Date of Birth" field to select your date of birth.

Figure 22

Jobseeker Profile Details



The screenshot shows a form titled "Profile details" with a subtitle "Update your profile information. Please note that this details will be shared with the recruiters." The form contains four input fields: "First name" (containing "John"), "Last name" (containing "Doe"), "Email address" (containing "j@gmail.com"), and "Date of Birth" (containing "01/01/2000" with a calendar icon). A "Save" button is located at the bottom right of the form.

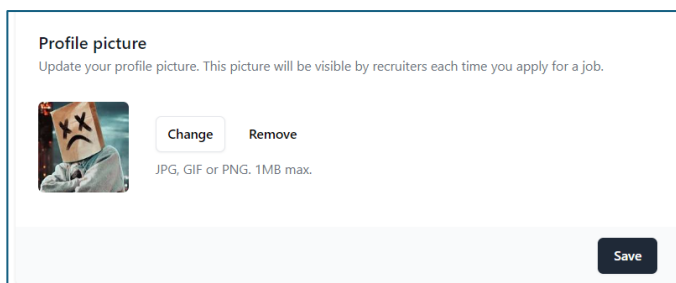
Jobseeker: Profile Picture

The Profile Picture Settings section allows the jobseeker to upload and update your profile picture, which will be visible to recruiters.

Click the "Upload" or "Change" button next to the profile picture placeholder, and then select an appropriate picture from your device, ensuring it is clear and professional. Supported file formats typically include JPEG, PNG, and GIF. Finally, click the "Save" button to update your profile picture.

Figure 23

Add, Change or Delete your Profile Picture



The screenshot shows a section titled "Profile picture" with a subtitle "Update your profile picture. This picture will be visible by recruiters each time you apply for a job." It features a placeholder image of a person with a mask. Below the image are two buttons: "Change" and "Remove". Below these buttons is the text "JPG, GIF or PNG. 1MB max." A "Save" button is located at the bottom right of the section.

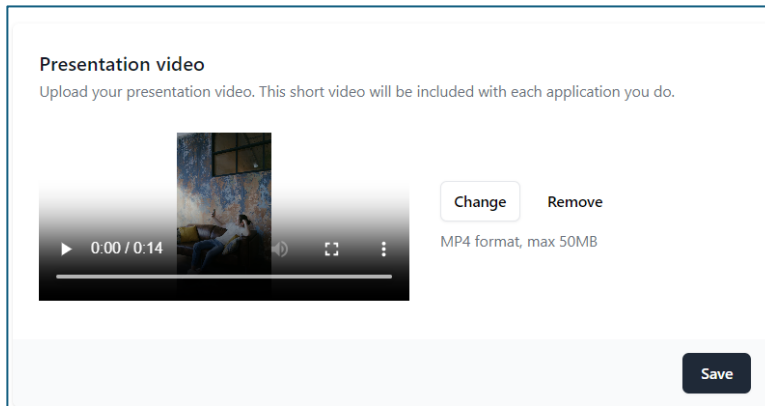
Jobseeker: Presentation Video

The Presentation Video section allows the jobseeker to upload a short video introducing yourself. This video will be included with each application you submit, providing recruiters with a personal

introduction. Navigate to the presentation video section within your profile settings. Click the "Change" button next to the video placeholder and select an appropriate MP4 video file from your device, ensuring the file size is within the 50MB limit. If you need to remove the current video, click the "Remove" button. After uploading your video, click the "Save" button to apply the changes.

Figure 24

Add, Change or Delete your Introductory Video

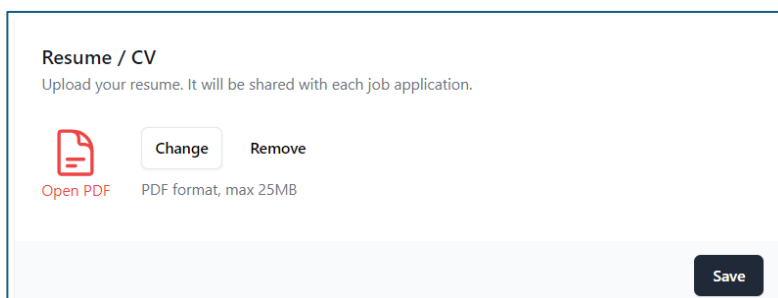


Jobseeker: Curriculum Vitae (CV)

The Resume / CV section allows the job seeker to upload their resume, which will be shared with each job application you submit. Follow the steps below to upload and manage your resume effectively. Navigate to the resume section within your profile settings and click the "Change" button beside the current resume placeholder. Select an appropriate PDF file from your device, ensuring the size is within the 25MB limit. If you need to remove the current resume, click the "Remove" button. Finally, click the "Save" button after uploading your resume to apply the changes.

Figure 25

Add, Change or Remove CV

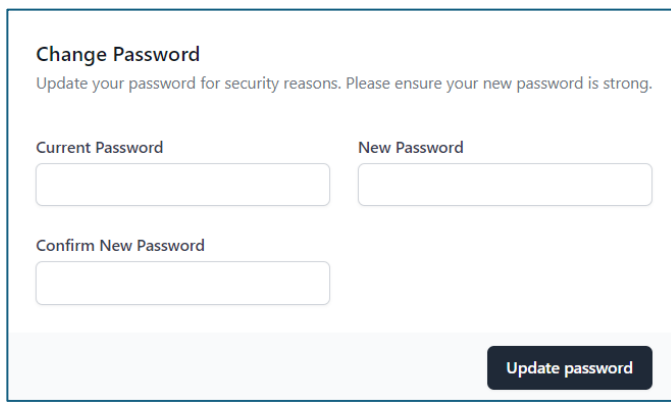


Change Password

The Change Password screen allows you to update your password for security reasons. First, enter your current password to change it. Next, type your new password, ensuring it is strong and meets the security requirements. Re-enter your new password to verify it. Finally, click the "Update password" button to save your new password.

Figure 26

Change Password

A screenshot of a web form titled "Change Password". Below the title is a subtitle: "Update your password for security reasons. Please ensure your new password is strong." The form contains three input fields: "Current Password", "New Password", and "Confirm New Password". The "Current Password" and "New Password" fields are side-by-side, while "Confirm New Password" is below them. At the bottom right of the form is a dark button labeled "Update password".

Change Password
Update your password for security reasons. Please ensure your new password is strong.

Current Password

New Password

Confirm New Password

Update password

Jobseeker: Job Openings

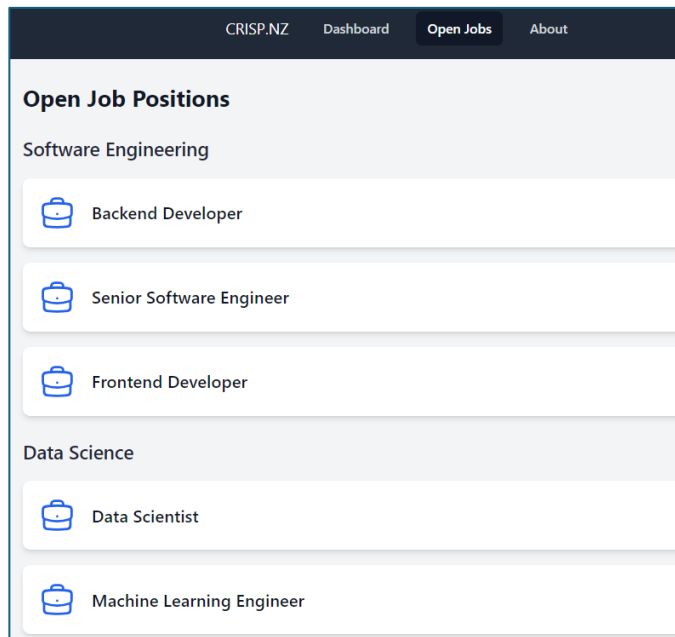
The Open Job Positions page allows jobseekers to browse and apply for available job openings.

This section is divided into categories to help you quickly find relevant positions.

Navigate to the Open Jobs page by clicking the "Open Jobs" tab in the top menu. The job positions are organised into categories, such as Software Engineering and Data Science. Scroll through the categories to find the type of job you are interested in. Within each category, you will see a list of available job positions. For example, you may find positions such as Backend Developer, Senior Software Engineer, and Frontend Developer under Software Engineering. At the same time, you might see positions like Data Scientist and Machine Learning Engineer under Data Science.

Figure 27

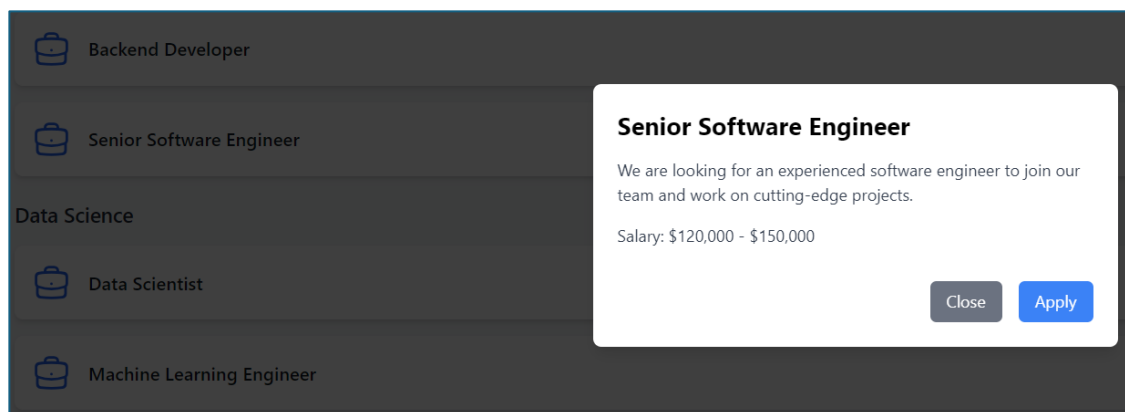
Open Job Positions.



Click on the job title for more details about the position, which typically includes a job description, requirements, responsibilities, and other relevant information. If you find a job that matches your skills and interests and is attractive, click the apply button.

Figure 28

View Job Details and Apply



Once you submit your application, you will receive a confirmation message that it was sent successfully.

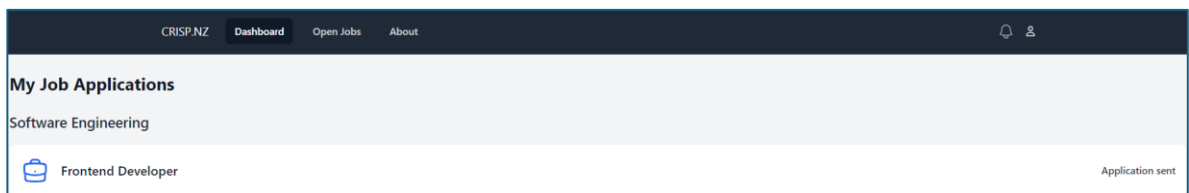
Jobseeker: My Job Applications

The jobseeker navigates to the My Job Applications page by clicking the "Dashboard" tab in the top menu.

The My Job Applications page lets you track the positions you have applied for. Job categories are organised in this section to help you easily manage your applications. Applications are listed under the relevant job categories. For example, if you have applied for a position in Software Engineering, it will be listed under that category.

Figure 29

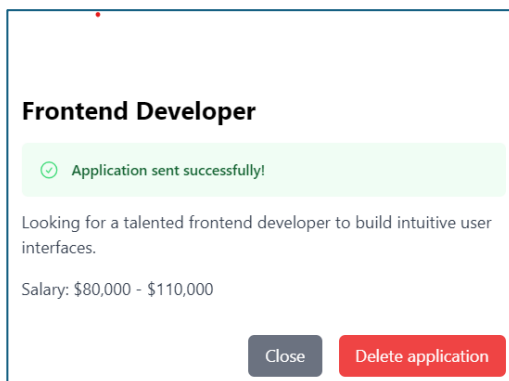
My Job Applications



To manage or view the details of your job applications, click on the Job Description. The screen displays information on the job position, including the job title, a brief job description, and the salary range. If you want to exit the confirmation screen and return to the previous page or your dashboard, click the "Close" button. If you wish to withdraw your application, click the "Delete application" button and confirm your decision in any subsequent prompts to remove your application from consideration.

Figure 30

Application Management Screen



Recruiter: Dashboard

The Recruiter Dashboard allows the recruiter to create job postings, view a list of current job postings, and see all the applicants.

To create a job posting, enter the job title, select the appropriate job category from the dropdown menu labelled "Category," enter the salary for the position, and provide a detailed job description. Once all fields are filled out, click the "Create Job" button to post the job.

The "Job Postings" section lists all your current job postings. The "Applicants" section lists applicants for your job postings.

Figure 31
Dashboard.

The screenshot displays the 'Recruiter Dashboard' interface. At the top, a dark navigation bar contains the text 'CRISP.NZ', 'Dashboard' (highlighted), 'Open Jobs', and 'About', along with a bell icon and a user profile icon. The main content area is titled 'Recruiter Dashboard' and is divided into three sections: 'Create Job Post', 'Job Postings', and 'Applicants'.

The 'Create Job Post' section on the left contains a form with the following fields: 'Job Title*' (text input), 'Category*' (dropdown menu with 'Select a category' selected), 'Salary*' (text input), and 'Job Description*' (text area). A 'Create Job' button is located at the bottom of this form.

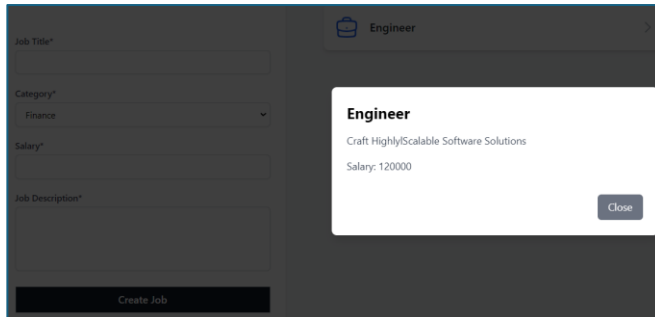
The 'Job Postings' section in the center lists several job postings, each with a brief icon and a title, followed by a right-pointing arrow: 'Expert in Sales', 'Programmer', 'Test', 'Expert in Sales', 'Data Analyst', 'Programmer', and 'Front-End Developer'.

The 'Applicants' section on the right lists two applicants, each with an email address and a right-pointing arrow: 'mck7147@autuni.ac.nz' and 'miscrits4321@gmail.com'.

Click on the job title for more details about a specific job posting.

Figure 32

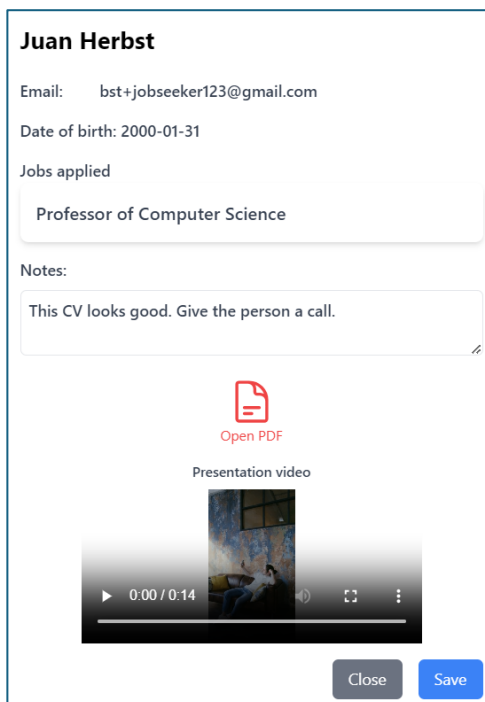
View Job Details



Click on an applicant to view more details about a specific applicant, including reading their CV, watching their Introductory Video or capturing notes on their profile.

Figure 33

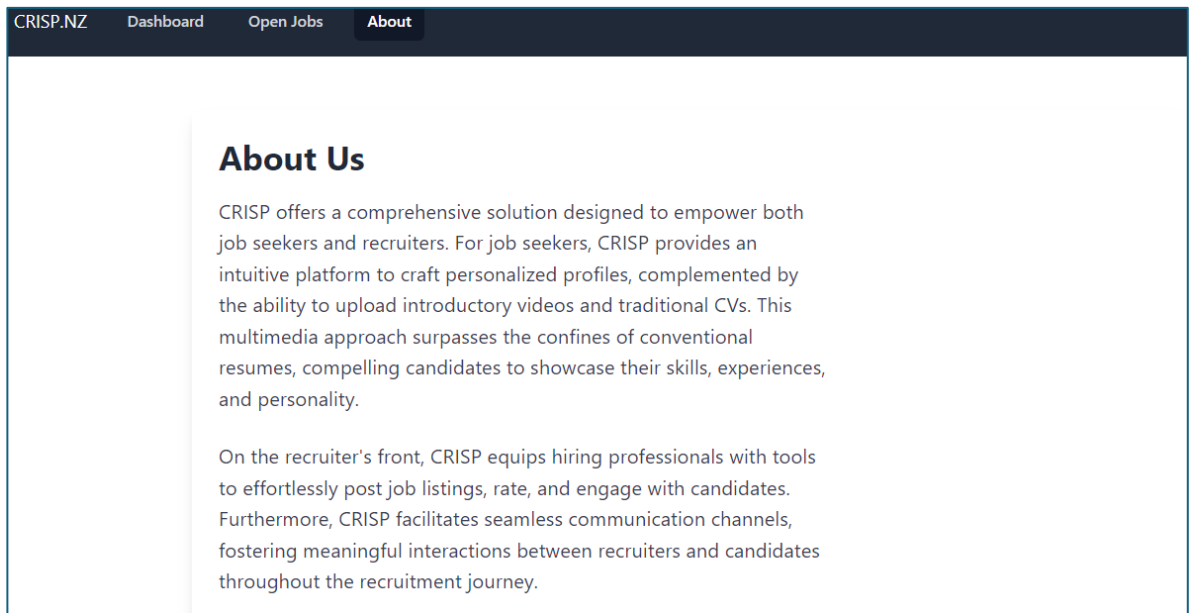
View Applicant Details, including CV and Presentation Video, and Add Private Notes.



About

The About Us section overviews Crisp and its mission to empower jobseekers and recruiters. Here, you can learn about the platform's features and how it facilitates recruitment.

Figure 34
About Crisp



Summary

The Crisp project is deployed on AWS, renowned for its cost-effectiveness, scalability, resilience, and availability. Crisp is a single-page application hosted in Amazon S3 and accessed via the custom domain www.crisp.nz. The backend adheres to the Platform as a Service (PaaS) delivery model and leverages serverless technology. The design, development, and quality assurance processes used Scrum as the agile project delivery methodology. Standard DevOps practices, including Amazon CloudWatch for logging and monitoring, were employed, with metrics sourced from native AWS services such as DynamoDB. Continuous Integration (CI) practices are implemented using Gitflow and GitHub for source control. In contrast, Continuous Deployment (CD) is achieved by merging code into the main branch through pull requests, ensuring code quality via the code review process.

Appendix

Group 8 Project Contributors & Source Control

Juan Herbst, Mike Chene, and Tommaso Cammelli jointly developed and realised the Crisp project. Their collective efforts, expertise, and synergy were instrumental in bringing this innovative initiative to fruition. Through their coordinated collaboration, they integrated their diverse skill sets and perspectives during the implementation of the Crisp project.

Table 11

Group 8 Contributors & Source Control Repository

Contributor	Student ID	GitHub Handle	Source Control Repository
Juan Herbst	13840146	jjherbst	https://github.com/tommcamm/aut-crisp
Mike Chene	19070279	mikeCHENE	
Tommaso Cammelli	23215488	tommcamm	

Total Expenditure

From March 2024 to May 2024, the AWS cost breakdown reveals total expenditures of approximately \$1.50 in March, \$0.75 in April, and \$0.75 in May. These costs are primarily driven by Amazon Route 53, which incurred charges of \$1.00 in March and \$0.50 in April and May and associated taxes amounting to \$0.50 in March and \$0.25 in April and May. This optimised cost structure is due to careful service consideration and architecting while staying within the parameters of the AWS free tier, which leads to significant cost savings.

Figure 35

Total Project Expenditure

