詳細設計書

4I 17 番 齋藤 健吾

2024 年 10 月 29 日

1　はじめに

本詳細設計書では、企業とエンジニアのジョブマッチングを円滑に行うためのシステムの詳細設計について記述する。機能一覧では、フロントエンドとバックエンドの機能をそれぞれ示し、入出力データ一覧では、各機能における入出力データを示す。また、このシステム詳細設計所の各機能における命名は、システム基本計画書にあるユースケース図・画面遷移図で示した語句と対応している。

2　機能一覧

表 1　機能一覧 (フロントエンド)

機能番号	機能名	処理番号	処理名	説明
1	ログインページ表示 (共通)	1-1	OAuth2.0 認証	Google でログイン ボタンを押すと、Google アカウントの選択画面を表示する
		1-2	既知のログイン判定	既にログインされていた場合、データを取得しログイン
		1-3	ロール[*1]選択	初回のログイン時に engineer / company を選択させる
		1-4	Submit ボタン	選択したロールと Google アカウントを DB に保存
2	各種情報表示[*2](共通)	2-1	アカウント情報表示	自身の Google アカウントを参照し、ログイン情報を表示する
		2-2	ログアウトボタン	押すとログアウト処理が行われるボタン
		2-3	戻るボタン	一つ前のページに戻るボタン
3	メニュー一覧表示 (エンジニア側)	3-1	一覧画面表示ボタン	企業一覧へ遷移するボタン
		3-2	プロフィール投稿/削除 遷移ボタン	プロフィール投稿/削除画面に遷移するボタン
		2-1	アカウント表示	同上
		2-2	ログアウトボタン	同上
4	メニュー一覧表示 (企業側)	4-1	一覧画面表示ボタン	エンジニア一覧と企業一覧へ遷移するボタン
		4-2	募集/削除 遷移ボタン	エンジニアへの募集投稿/削除画面に遷移するボタン
		2-1	アカウント表示	同上
		2-2	ログアウトボタン	同上
5	企業一覧表示 (エンジニア/企業側)	5-1	企業一覧取得	Firebase Realtime DB から企業一覧を取得し、会社名や募集内容を表示する
		5-2	企業詳細表示	クリックした際、ポップアップで企業の詳細情報を表示する
		2-1	アカウント表示	同上
		2-2	ログアウトボタン	同上
		2-3	戻る:width ボタン	同上
6	エンジニア一覧表示 (企業側)	6-1	エンジニア一覧取得	Firebase Realtime DB からエンジニア一覧を取得し、名前や応募内容を表示する
		6-2	エンジニア詳細表示	クリックした際、ポップアップでエンジニアの詳細情報を表示する
		2-1	アカウント表示	同上
		2-2	ログアウトボタン	同上
		2-3	戻るボタン	同上
7	プロフィール投稿/削除 (エンジニア側)	7-1	名前入力	エンジニアの名前を入力するフォーム
		7-2	応募内容入力	エンジニアの応募内容を入力するフォーム
		7-3	投稿ボタン	上記 (7-1,7-2) で入力した情報を Firebase Realtime DB に保存するボタン
		7-4	削除ボタン	自身が以前応募した情報がある場合、その応募を削除するボタン
		2-1	アカウント表示	同上
		2-2	ログアウトボタン	同上
		2-3	戻るボタン	同上
8	募集投稿/削除 (企業側)	8-1	会社名入力	企業の会社名を入力するフォーム
		8-2	募集内容入力	企業の募集内容を入力するフォーム
		8-3	投稿ボタン	上記 (8-1, 8-2) で入力した情報を Firebase Realtime DB に保存するボタン
		8-4	削除ボタン	自身が以前募集した情報がある場合、その募集を削除するボタン
		2-1	アカウント表示	同上
		2-2	ログアウトボタン	同上
		2-3	戻るボタン	同上
9	結果表示 (エンジニア側)	9-1	マッチ度計算ボタン	Firebase Cloud Functions のエンドポイントを、自身の情報を載せて叩く
		9-2	結果表示	上記ボタン (9-1) を押した後、その返答結果をマッチ度の降順に表示する
		9-3	企業詳細表示	クリックした際、ポップアップで企業の詳細情報を表示する
		2-1	アカウント表示	同上
		2-2	ログアウトボタン	同上
		2-3	戻るボタン	同上
10	結果表示 (企業側)	10-1	マッチ度計算ボタン	Firebase Cloud Functions のエンドポイントを、自身の情報を載せて叩く
		10-2	結果表示	上記ボタン (10-1) を押した後、その返答結果をマッチ度の降順に表示する
		10-3	エンジニア詳細表示	クリックした際、ポップアップでエンジニアの詳細情報を表示する
		2-1	アカウント表示	同上
		2-2	ログアウトボタン	同上
		2-3	戻るボタン	同上

[*1] ロール: ユーザに割り当てられる役割。エンジニアか企業かのいずれかである。

[*2] ログインページ以外に繰り返し用いられる機能・コンポーネント。2 回目以降の説明は省略し、灰色で示している。

表 2　機能一覧 (バックエンド)

機能番号	機能名	処理番号	処理名	説明
11	エンドポイントの常駐	11-1	エンドポイントの確立	Firebase Cloud Functions に findmatches エンドポイントを立てる
		11-2	エンドポイントの処理	エンドポイントにリクエストが来た際、下記 (12, 13, 14) を用いて処理を行う
		11-3	結果の返答	処理結果を POST 元に返答する
12	データベースでの情報取得	12-1	DB への通信の確立	Firebase Realtime Database へ Credential(json) を用いて認証を行い、読み書きを可能にする
		12-2	ユーザ[*3]情報取得	ユーザ (一人) の情報をデータベースから取得し呼び出し元へ返す
		12-3	ターゲット[*4]情報取得	ターゲット (複数人) の情報をデータベースから取得し呼び出し元へ返す
13	マッチング処理	13-1	マッチング処理	上記処理 (12) で取得した情報を入力とし、下記 (14) を用いて TF-IDF を計算する
		13-2	結果の整形	下記 (14) で計算された TF-IDF 行列から、ターゲットを Cosine 類似度[*5]の降順にソートする
		13-3	結果の返答	マッチング結果 (13-2) をを呼び出し元に返す
14	ベクトル化 TF-IDF 行列計算	14-1	テキスト分割	テキストを janome で形態素解析し、分割する
		14-2	ベクトル・TF-IDF 計算	分割したテキストをベクトル化し、sklearn で TF-IDF を計算する
		14-3	結果の返答	結果である TF-IDF 行列 を呼び出し元へ返す

機能一覧を表 1, 2 に示す。

表 1 はフロントエンドの機能一覧であり、ログインページ表示、各種情報表示、メニュー一覧表示、企業一覧表示、エンジニア一覧表示、プロフィール投稿/削除、募集投稿/削除、結果表示の機能がある。これらの機能は、フロントエンドのコンポーネントとして実装され、実装には Next.js を用いて、デプロイには Vercel を用いる。また、データの取得には Firebase Realtime Database を用いる。

表 2 はバックエンドの機能一覧であり、エンドポイントの常駐、データベースでの情報取得、マッチング処理、ベクトル化・TF-IDF 計算 の機能がある。これらの機能は、バックエンドのコンポーネントとして実装され、実装には Python を用いて、デプロイには Firebase Cloud Functions を用いる。また、データの取得には Firebase Realtime Database を用いる。

[*3] ユーザ: マッチ度計算ボタンを押したクライアントのこと。1 人。

[*4] ターゲット: ユーザに対し、マッチングを行う対象。すなわちユーザとは異なるロールを持つ人の集合のこと。複数人。

[*5] Cosine 類似度: ベクトル空間モデルにおいて、ベクトル間の類似度を測る指標。ここでは、TF-IDF ベクトルの類似度を測る。

3　入出力データ一覧

表 3　入出力データ一覧 (フロントエンド)

処理番号	機能名	データ番号	データ名	詳細 (: 型)・制約
1-1	OAuth2.0 認証	1-1-1	Google アカウント選択	OAuth 画面からの Google アカウントの選択 (:any)
		1-1-2	Google アカウント情報	Google アカウントの情報取得 (OAuth からの token:String)
1-2	既知のログイン判定	1-2-1	ログイン情報	以前のログイン情報の有無を判定 (isFirstLogin:Boolean)
		1-2-2	ログイン情報	!isFirstLogin => ログイン情報の取得 (userInfo:{”name”:String, ”email”:String, ”role”:String, . . . })
1-3	ロール選択	1-3-1	ロール選択	isFirstLogin => ロールを engineer/company から選択 (role:String)
1-4	Submit ボタン	1-4-1	ログイン情報	1-3-1 => ログイン情報の保存・取得 (userInfo:{”name”:String, ”email”:String, ”role”:String, . . . })
2-1	アカウント情報表示	2-1-1	ログイン情報	ログイン情報の取得 (userInfo[”name”]:String, userInfo[”email”]:String, userInfo[”role”]:String)
2-2	ログアウトボタン	2-2-1	ログイン情報	ログアウト処理の実行・ログイン情報の削除 (handleLogout => setUserInfo(null);)
2-3	戻るボタン	2-3-1	Router	一つ前のページに戻る (onclick => router.back();)
3-1	一覧画面表示ボタン	3-1-1	Router	企業一覧画面に遷移 (onClick => router.push(”/list?role=engineer”))
3-2	プロフィール投稿/削除 遷移ボタン	3-2-1	Router	プロフィール投稿/削除画面に遷移 (onClick => router.push(”/registrate”))
4-1	一覧画面表示ボタン	4-1-1	Router	エンジニア一覧画面に遷移 (onClick => router.push(”/list?role=company”))
		4-1-2	Router	企業一覧画面に遷移 (onClick => router.push(”/list?role=engineer”))
4-2	募集/削除 遷移ボタン	4-2-1	Router	募集/削除画面に遷移 (onClick => router.push(”/register”))
5-1	企業一覧取得	5-1-1	全企業情報	企業一覧として一部分を取得 (:{{企業名:item[”name”]:String, 募集内容:item[”description”]:String}, . . . })
5-2	企業詳細表示	5-2-1	企業詳細情報	企業の詳細情報を表示 (:{(上記 5-1-1 に加え), メール:item[”email”]:String, 投稿日時:item[”date”]. . . })
6-1	エンジニア一覧取得	6-1-1	全エンジニア情報	エンジニア一覧として一部分を取得 (:{{名前:item[”name”]:String, 応募内容:item[”description”]:String}, . . . })
6-2	エンジニア詳細表示	6-2-1	エンジニア詳細情報	エンジニアの詳細情報を表示 (:{(上記 6-1-1 に加え), メール:item[”email”], 投稿日時:item[”date”], . . . })
7-1	名前入力	7-1-1	名前	エンジニアの名前を入力するフォーム (name:String)
7-2	応募内容入力	7-2-1	応募内容	エンジニアの応募内容を入力するフォーム (description:String)
7-3	投稿ボタン	7-3-1	投稿内容	上記 (7-1,7-2) で入力した情報を保存 (handleSubmit => set(DBRef, {”name”:name, ”description”:. . . , . . . }))
7-4	削除ボタン	7-4-1	削除内容	以前の応募内容を削除 (handleDelete => remove(dataRef))
8-1	会社名入力	8-1-1	会社名	企業の会社名を入力するフォーム (name:String)
8-2	募集内容入力	8-2-1	募集内容	企業の募集内容を入力するフォーム (description:String)
8-3	投稿ボタン	8-3-1	投稿内容	上記 (8-1,8-2) で入力した情報を保存 (handleSubmit => set(DBRef, {”name”:name, ”description”:. . . , . . . }))
8-4	削除ボタン	8-4-1	削除内容	以前の募集内容を削除 (handleDelete => remove(dataRef))
9-1	マッチ度計算ボタン	9-1-1	ユーザ情報	ユーザの情報をエンドポイントに送信 (onClick => fetch(”https. . . /findmatches”, body:{role: role, email: email}))
9-2	結果表示	9-2-1	マッチング結果	エンドポイントからの返答を受け取る (:[{email:String, score:int}, . . .]), これから募集内容を取得・表示 (:(5-1-1 の型))
9-3	企業詳細表示	9-3-1	企業詳細情報	企業の詳細情報を表示 (:{(上記 9-2-1 に加え), メール:item[”email”]:String, 投稿日時:item[”date”]. . . })
10-1	マッチ度計算ボタン	10-1-1	ユーザ情報	ユーザの情報をエンドポイントに送信 (onClick => fetch(”https. . . /findmatches”, body:{role: role, email: email}))
10-2	結果表示	10-2-1	マッチング結果	エンドポイントからの返答を受け取る (:[{email:String, score:int}, . . .]), これから応募内容を取得・表示 (:(6-1-1 の型))
10-3	エンジニア詳細表示	10-3-1	エンジニア詳細情報	エンジニアの詳細情報を表示 (:{(上記 10-2-1 に加え), メール:item[”email”], 投稿日時:item[”date”], . . . })

表 4　入出力データ一覧 (バックエンド)

処理番号	機能名	データ番号	データ名	詳細 (: 型)・制約
11-1	エンドポイントの確立	11-1-1	エンドポイント	Cloud Functions に findmatches エンドポイントを立てる (req:{role:String, email:String}:https_fn.Request → https_fn.Response)
11-2	エンドポイントの処理	11-2-1	リクエスト	エンドポイントにリクエストが来た際の情報を受け取る (req: {role:String, email:String}:https_fn.Request)
		11-2-2	レスポンス	処理結果を呼び出し元へ返答 (res: [{email:String, score:int}, . . .]:https_fn.Response)
12-1	DB への通信の確立	12-1-1	DB 情報	Firebase Realtime Database への Credential(:json)
12-2	ユーザ情報取得	12-2-1	ユーザ情報	ユーザの情報を取得 (:{email:String, discription:String})
12-3	ターゲット情報取得	12-3-1	ターゲット情報	ターゲットの情報を取得 (:[{email:String, discription:String}, . . .])
13-1	マッチング処理	13-1-1	ユーザ・ターゲット文章	ユーザとターゲットの文章を入力とし、下記函数 (14) へ渡す (:{discriptionUser:String, [discriptionTarget:String, . . .]})
13-2	結果の整形	13-2-1	マッチング結果	下記函数 (14) からのマッチング結果を整形 (:[[float, float, . . .], [float, float, . . .], . . .]$^\top \in \mathbb{R}^{n\times m}$) → ([{email:String, score:int}, . . .])
13-3	結果の返答	13-3-1	マッチング結果	スコアの降順になったマッチング結果を呼び出し元へ返答 (:[{email:String, score:int}, . . .])
14-1	テキスト分割	14-1-1	分割テキスト	受け取ったテキストそれぞれを janome で形態素解析し、分割 (:discription:String → discription:[String, String, . . .])
14-2	ベクトル・TF-IDF 計算	14-2-1	ベクトル	テキストをベクトル化 (:[String, String, . . .] → [float, float, . . .])
		14-2-2	TF-IDF 行列	ベクトルを用いて TF-IDF 行列を計算 (:[[float, float, . . .], [float, float, . . .], . . .]$^\top \in \mathbb{R}^{n\times m}$)
14-3	結果の返答	14-3-1	TF-IDF 行列	TF-IDF 行列を呼び出し元へ返答 (:[[float, float, . . .], [float, float, . . .], . . .]$^\top \in \mathbb{R}^{n\times m}$)

入出力データ一覧を表 3, 4 に示す。

表 3 はフロントエンドの入出力データ一覧であり、OAuth2.0 認証、既知のログイン判定、ロール選択、アカウント情報表示、ログアウトボタン、戻るボタン、遷移ボタン、相手の投稿の一覧取得、投稿内容入力、投稿ボタン、削除ボタン、マッチ度計算ボタン、結果表示、詳細表示 等の入出力データがある。

表 4 はバックエンドの入出力データ一覧であり、エンドポイントの確立、エンドポイントの処理、DB への通信の確立、ユーザ情報取得、ターゲット情報取得、マッチング処理、結果の整形、テキスト分割、ベクトル・TF-IDF 計算、結果の返答の入出力データがある。

全体像として、フロントエンドではユーザの操作によってデータを取得・表示する機能があり、バックエンドではフロントエンドからのリクエストに対してデータを取得・計算し、結果を返答する機能がある。また、このように分けた理由は、tokenize(vectolize) や TF-IDF などの自然言語処理は、Python で実装するのが一般的であるためである。さらに、データベースへのアクセスは、Firebase Realtime Database が提供する REST API を用いることで、フロントエンドから直接アクセスすることができるため、バックエンドへのデータの受け渡しを直接行うことなく、データの取得・保存を行うことができる。これによって、フロント・バックエンド間のデータの受け渡しを簡略化し、データの取得・保存の効率化を図っている。