جامعة أم القرى
UMM AL-QURA UNIVERSITY

شاغر

Software Engineering

14013303-3

SUPERVISOR:
DR. AREEJ ALFERAIH

Chapter 1

Requirement Specification

03

Introduction

1.1 Problem Definition

In today's competitive job market, institutions, job seekers, and university students looking for summer training face several challenges. Institutions struggle to find suitable candidates for open positions and internships, while job seekers and students find it difficult to discover relevant opportunities in their fields. The process of connecting applicants with institutions is often inefficient and time-consuming, leading to missed opportunities and increased frustration for all parties involved.

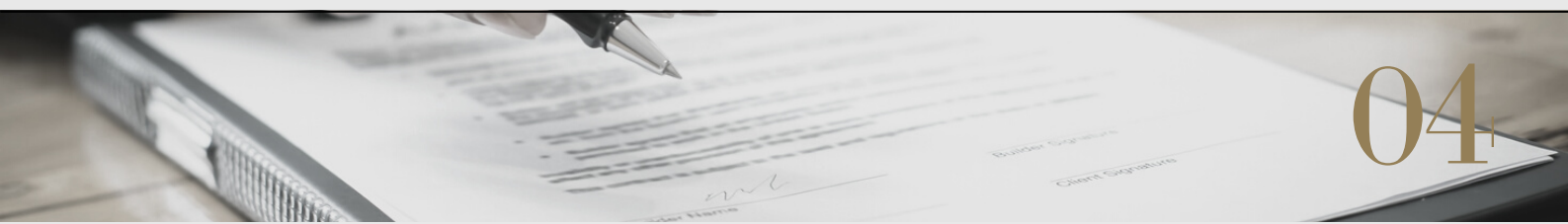
1.2 Solutions

Our software engineering project aims to address these challenges by developing a comprehensive application that serves as a bridge between institutions, job seekers, and university students seeking summer training. The platform will enable users to create detailed profiles, including personal information and skill sets, while institutions can post job advertisements and internship opportunities. This centralized platform will facilitate more efficient and effective connections between applicants and institutions, streamlining the process and improving overall satisfaction.

1.3 Aims and objectives

The primary aims and objectives of this project are as follows:

- To create a user-friendly application that simplifies the process of connecting institutions with job seekers and students seeking summer training.
- To provide a platform for job seekers and students to showcase their skills and qualifications in a professional and organized manner.
- To enable institutions to efficiently post job advertisements and internship opportunities, reaching a wider audience of potential candidates.
- To implement advanced search and matching algorithms that help users discover relevant opportunities and institutions find suitable candidates.



Functional requirements

SIGN UP:

USER REQUIREMENTS:

1: The user shall be able to create a new account.

SYSTEM REQUIREMENTS:

1.1: If the user clicks on the "**Create account**" button, they will go to the account creation page.

1.2: The system shall ask the user to choose if they are a job/training seekers or an institution.

1.3: The system shall display a sign up form which contains a list of fields :
(first name, last name, email, confirmation of the email, password, confirmation of the password, phone number)

1.4: The system shall validate the entered data with these constrains:

- The email and phone number must be unique and there is no other record in the database has the same values.
- The password should be at least 8 characters that contains letters in uppercase and lowercase and numbers.
- The email and the confirmed email must be identical.

1.5: When the user clicks on "**sign up**" button and the entered data were valid, the account is created and the system goes to the login page.

1.6: The system shall sava these information to the "***user_info***" table in the database.

Functional requirements

LOG IN:

USER REQUIREMENTS:

2: The user shall be able to log in.

SYSTEM REQUIREMENTS:

2.1: The system shall display login form to the user which contain these fields: (email and password).

2.2: The system shall check and compare the entered data with the "*user_info*" table in the database. if there is a match retrieve user information and navigate to the main interface. otherwise, display pop-up message contains "**there is no such account, try again or sign up for a new account**".

Functional requirements

BUILD CV:

USER REQUIREMENTS:

3: the applicants shall be able to build their cv or or upload a previous CV file

SYSTEM REQUIREMENTS:

3.1: The system shall provide a "**Build my CV**" button in the homepage. when the user click on it, the system shall display a form page with multiple fields.

3.2: The system shall receive the CV files that the user uploads to the system

3.3: The system shall fill by default the applicants name, email and phone number from the "***user_info***" table in the database.

3.4: The system shall provide a drop-down list for the following information: (day/month/year of birth, nationality, gender, marital status, city of residence, major, degree, experience)

3.5: The system shall ask the user to write the following information: (national ID, GPA, employment history, achievements, hobbies).

3.6: The system shall set the choices for the "degree" field are (school, associate, bachelor's, graduate degree, doctorate, professional degree or none)

3.7: The system shall ask the user to attach their certifications.

3.8: The system shall ask the user to write more about their selves (optional).

3.9: At the end of the page the system shall provide a "**build CV**" button that saves all the information above to the "***userCV***" table in the database.

Functional requirements

SEE SUGGESTED JOB ADVERTISEMENTS:

USER REQUIREMENTS:

4: The applicants shall be able to see suggested job advertisements (depending on their qualification).

SYSTEM REQUIREMENTS:

4.1: If the applicant did not yet build their CV, The system shall choose a random job advertisements from the database and present them as boxes in scrolling way at the homepage.

4.2: If the applicant did build their CV, the system shall take the (nationality, gender, marital status, city of residence, major, degree, experience) information from the "***userCV***" table in the database and compare it with all the job advertisements' requirements from "***advertisement***" table in the database and present them as boxes in scrolling way at the homepage sorted by the most matching.

4.3: The system shall allow the user to click on the suggested job advertisements to see more details.

Functional requirements

ADDING POSTS:

USER REQUIREMENTS:

5: The user shall be able to add a new post.

SYSTEM REQUIREMENTS:

5.1: In the profile page, the system shall provide an “**add a new post**” button.

5.2: If the user click on this button, the system shall open a window with a textbox, “**attach file**” button, ”**insert photo/video**” button.

5.3: If the user click on “**attach file**” button, the system shall open the files app so the user can choose a file to attach.

5.4: If the user click on “**insert photo/video**” button, the system shall open the photos/videos gallery or studio for the user to insert.

5.5: The system shall provide a “submit” button that saves the text, files, photos, videos inserted to the "***user_posts***" table in the database and show it in the user’s profile.

5.6: If the user didn’t input a text or photo or video or file, the system shall not allow them to click on the “**submit**” button.

Functional requirements

FAVORITE LIST:

USER REQUIREMENTS:

6: The applicants shall be able to view their favorite list and add/delete from it.

SYSTEM REQUIREMENTS:

6.1: The system should have a user-friendly interface that allows users to easily create, view, and manage their favorite list.

6.2: The system shall allow users to add an Institution to their favorite list by clicking on a "**Add to Favorite**" button next to the institution profile then the system shall save the institution's name to the "***user_info***" table in the database.

6.3: The system shall allow the user to delete items from their favorite list by clicking on a "**remove from favorite**" button and the system shall delete the institution's name from the "**user_info**" table from the database.

6.4: The system shall provide a "**favorite**" button in the homepage. when the user clicks on it, it should generate a list of all the favorite institutions names based on the "***user_info***" table in the database.

6.5: If the user clicks on an institution's name, the system shall display the institution's profile.

Functional requirements

APPLY FOR A JOB/TRAINING:

USER REQUIREMENTS:

7: The applicants shall be able to apply for a job/training.

SYSTEM REQUIREMENTS:

7.1: The system shall provide an "**apply**" button next to every job/training advertisement.

7.2: When user click on the apply button, the system shall Compare the advertisement's requirements with the user qualifications from the "***userCV***" table and the "***advertisement***" table in the database.

7.3: If the user does not fit all the requirements, the system shall show this message: "**you are not qualified for this job/training**".

7.4: If the user fits all the requirements, the system will add the user CV to the "***applications_list***" table in the database.

7.5: The system shall show "**the application is sent**" message.

Functional requirements

SEARCH FOR JOBS:

USER REQUIREMENTS:

8: The applicants shall be able to search for jobs

SYSTEM REQUIREMENTS:

8.1: The system shall show a text field with a button "**search**" to search for a job/training in the homepage.

8.2: The system shall allow the user to search by either the company name or a specific job advertisement.

8.3: When the search button is pressed, The System shall compare between the user's searched text and the job advertisements from the "***user_info***" and "***advertisement***" tables in the database.

8.4: The system shall display the results in scrolling page (could be company names or job advertisements).

8.5: If the user click on a company's name, the system shall open a page with the company's name, Email and all their posts. the system shall get these Information from "***user_info***" and "***user_posts***" tables in the database.

8.6: If There is no similarity between the user's search and the database, the system shall Displays a "**No results found**" message .

Functional requirements

SEE APPLICATIONS STATUS:

USER REQUIREMENTS:

9: The applicants shall be able to see their applications status: (processing / accepted / rejected).

SYSTEM REQUIREMENTS:

9.1: The system should provide a "**my applications**" button in the homepage. when the user click on it, the system shall display a list (from the "***application_list***" table from the database) of all the advertisements' names that the user had applied for. beside each of them, the system should show a label with the current status of the application, whether it is waiting, accepted, or rejected by the company. the system shall bring this information from "**application_list**" table in the database.

9.2: The system should have a notification system that alerts applicants when there is a change in their application status.

Functional requirements

SEE JOB OFFERS:

USER REQUIREMENTS:

10: The applicants shall be able to see their job offers.

SYSTEM REQUIREMENTS:

10.1: The system shall provide a "**my job offers**" button on the homepage. when the applicant clicks on it, the system shall list all the job offer messages based on the "**job_offers**" table in the database.

10.2: The system shall provide the institution's Email (from the "***user_info***" table in the database) so that the user can contact them if they were interested.

Functional requirements

ADD A NEW ADVERTISEMENT:

USER REQUIREMENTS:

11: The institution shall be able to add a new advertisement (job/training).

SYSTEM REQUIREMENTS:

11.1: The system shall provide a "**new advertisement**" button in the homepage. when the user click on it, the system shall display a form page with multiple text fields.

11.2: The system shall fill by default the institution's name and email from the "***user_info***" table in the database.

11.3: The system shall provide a drop-down list for the following information: (day/month/year of commencement, job location, required gender, required major, required degree, required experience, required range of ages)

11.4: The system shall ask the user to write the following information: (job/training name, additional job requirement/details (if exists)).

11.5: The system shall ask the institution to choose the day, month and year advertisement start/end date.

11.6: When the user clicks on "send" button, the system shall save all of the advertisement's details into the "***advertisement***" table in the database.

11.7: The system shall show the advertisement in the institution’s profile.

Functional requirements

SEE JOB APPLICATIONS:

USER REQUIREMENTS:

12: The institution shall be able to see all the job applications and accept/reject it.

SYSTEM REQUIREMENTS:

12.1: The system shall provide a "**job applications**" button on the homepage. when the user clicks on it, the system shall generate a list of all the job/training advertisement's names from the "advertisement" table in the database. when the user clicks on one, the system shall generate a list of all the applicants' names for that specific job/training from the "***application_list***" table.

12.2: If the user clicks on an applicant's name, the system shall open a window with the applicant's CV from the "***userCV***" table in the database.

12.3: The system shall provide "**accept**" or "**reject**" buttons next to each application and the Users can click on only one of them.

12.4: The system shall save the application status "**accepted**", "**rejected**" or "**processing**" (if none had chosen) to the "***application_list***" table in the database.

12.5: The system shall provide a "**view profile**" button next to each applicant's name. when the user clicks it the system shall open the applicant's profile page.

12.6: When opening their profile page, The system shall be able to retrieve all applicants' profile information from the "***user_posts***" table from the database.

Functional requirements

SENDING JOB OFFERS

USER REQUIREMENTS:

13: The institution shall be able to send a job offer.

SYSTEM REQUIREMENTS:

13.1: The system shall provide an "offer jobs" button on the institution's homepage. when the user clicks on it, the system shall provide a list of users' names who are fitted to one of the job advertisements' requirements provided by this institution based on the "***advertisement***" table and "***userCV***" table from the database.

13.2: If the user clicks on a name, the system shall open a window with the applicant's CV from the "***userCV***" table in the database.

13.3: The system shall provide a "**view profile**" button next to each name. when the user clicks on it the system shall open the user's profile page.

13.4: The system shall provide an "**offer job**" button next to each name. when clicking on it, the system shall open a window with a text box for the institution to write their job offering message. then they shall press the "**send**" button that will save the offering message to the "***job_offers***" table in the database.

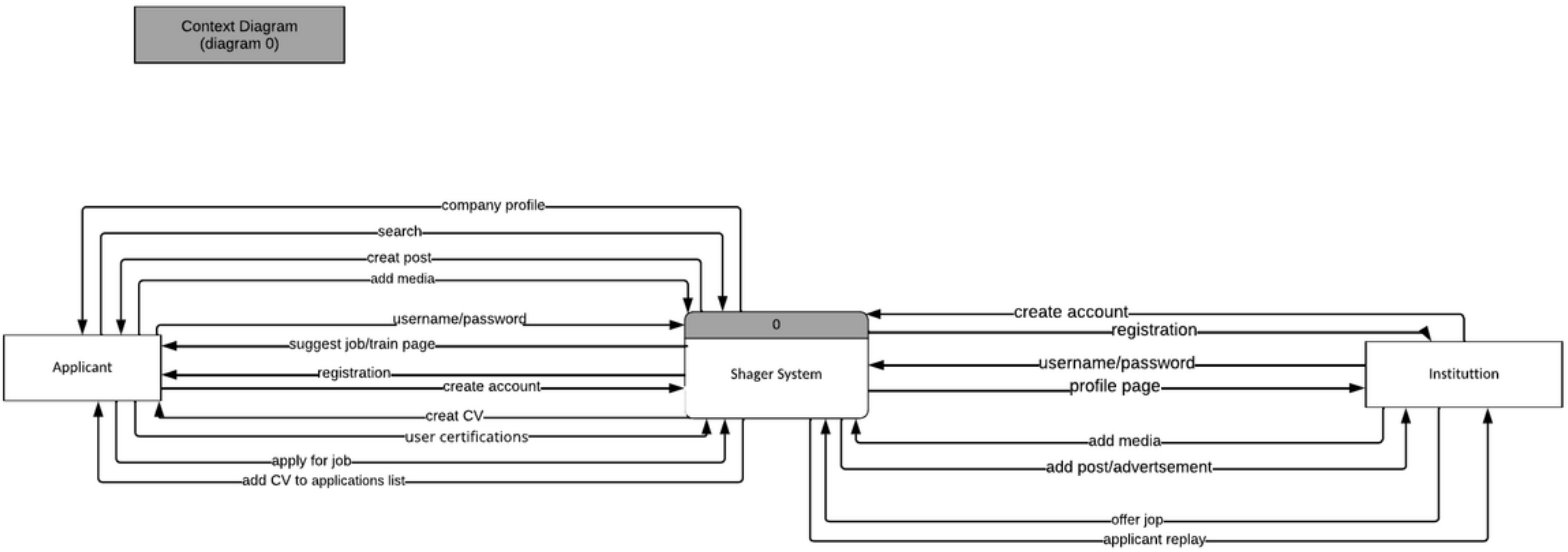
Non-Functional requirements

Usability	• The application should be easy to use, it will take the user 3 hours to learn how to use and the graphics design is clear.
Availability	• Our web must be available for the users 99% of the time per month. after that, it should be available for the users 99.998% of the time.
Efficiency (speed)	• Each page must load under 2 seconds. • The response time must be 1 second (A response time of about 1 seconds offers users an instant response, with no interruption)
Reliability	• The system must have a large database to handle a large number of users above 500 users. • Users can access their pages 98% of the time without failure.
Security	• The application should have robust security measures (confirmation of The user's password and email) in place to protect user data and prevent unauthorized access.

USE CASE

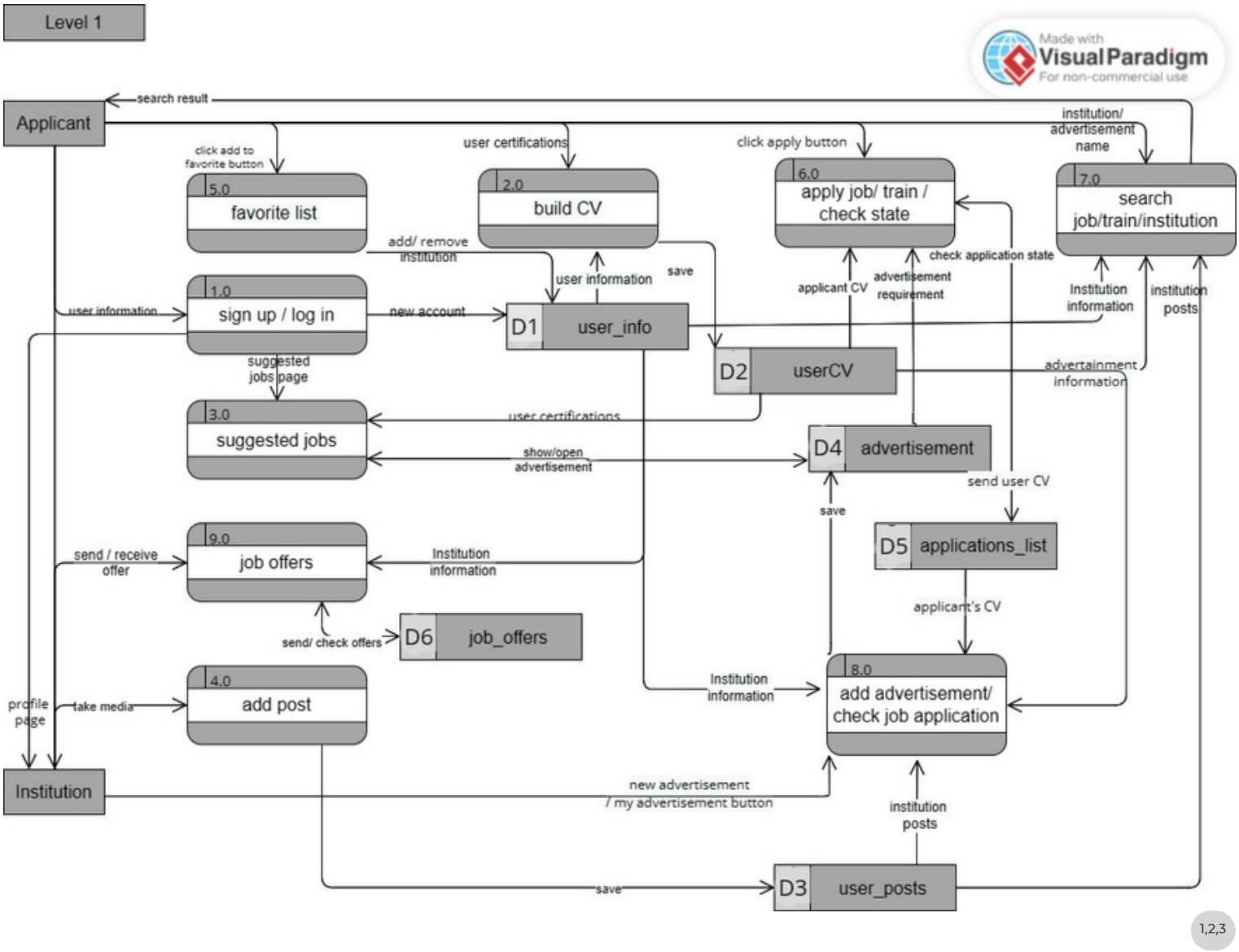

comfirmation password
<<include>>
Create Account
Log in
Build CV
View favorite list
Add advertisement
Apply job/training.
<<extends>>
<<extends>>
see suggested jobs
send job offer
Search jobs
View applications
View job offers
See application status
add post
INSTITUTION
APPLICANT

DFD diagram



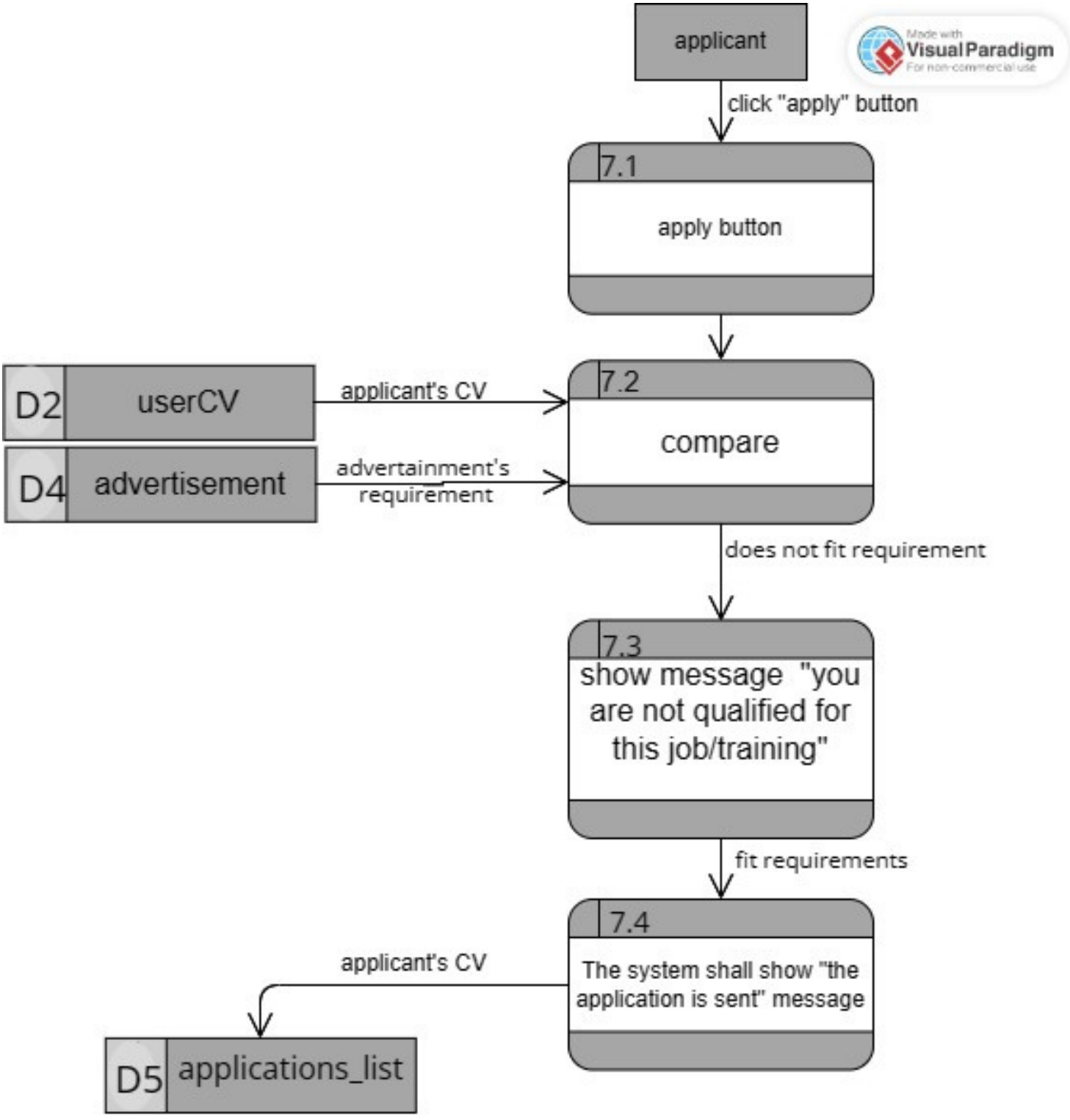
1,2,3

DFD diagram


Level 1
Made with VisualParadigm For non-commercial use
Applicant
search result
click add to favorite button
user certifications
click apply button
institution/ advertisement name
5.0 favorite list
2.0 build CV
6.0 apply job/ train / check state
7.0 search job/train/institution
add/ remove institution
user information
save
check application state
applicant CV
advertisement requirement
Institution information
institution posts
user information
1.0 sign up / log in
new account
D1 user_info
D2 userCV
suggested jobs page
advertainment information
3.0 suggested jobs
user certifications
show/open advertisement
D4 advertisement
send user CV
save
send / receive offer
9.0 job offers
Institution information
D5 applications_list
applicant's CV
send/ check offers
D6 job_offers
profile page
take media
4.0 add post
Institution information
8.0 add advertisement/ check job application
Institution
new advertisement / my advertisement button
institution posts
save
D3 user_posts
1,2,3

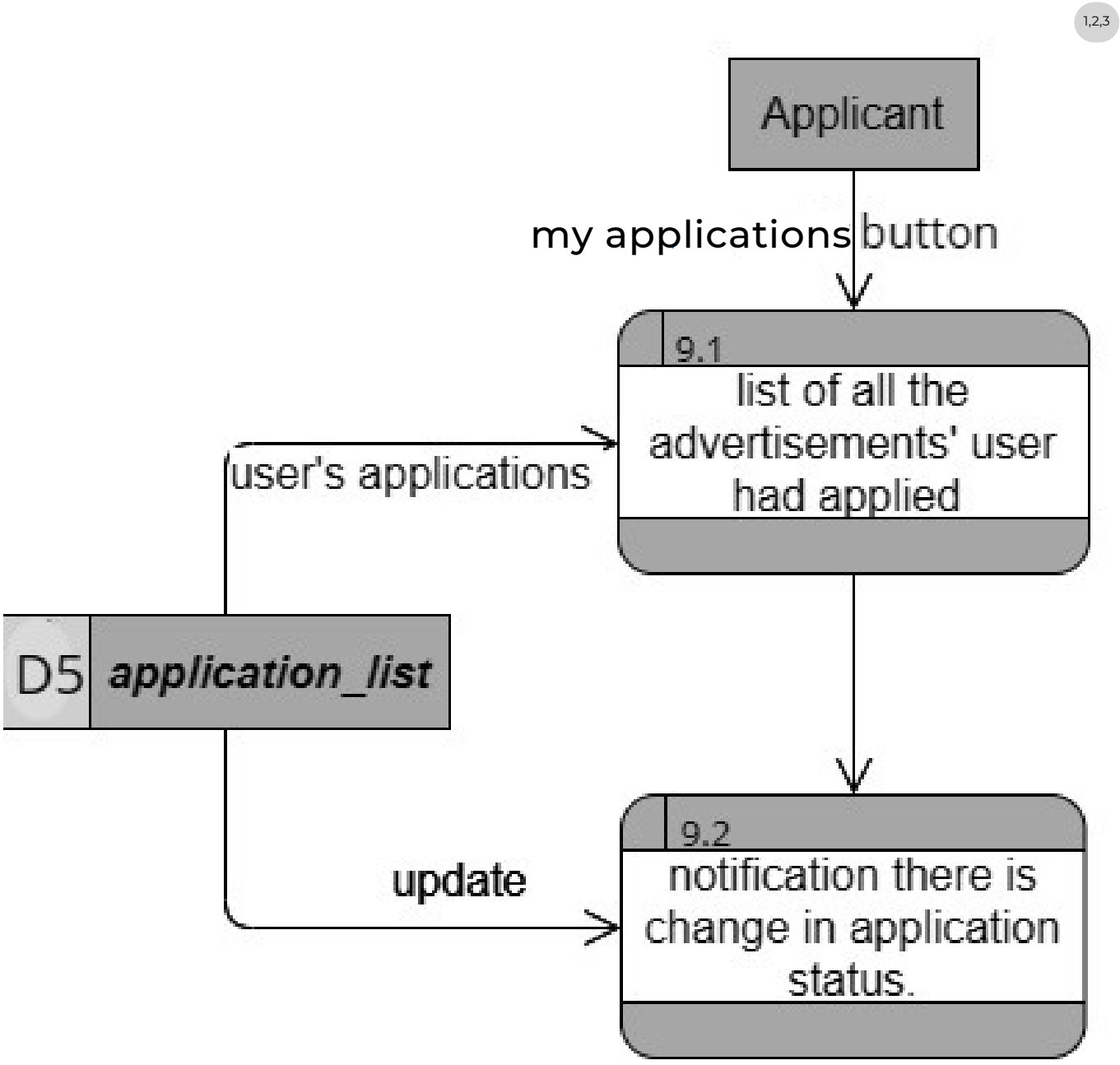
DFD diagram

level 2

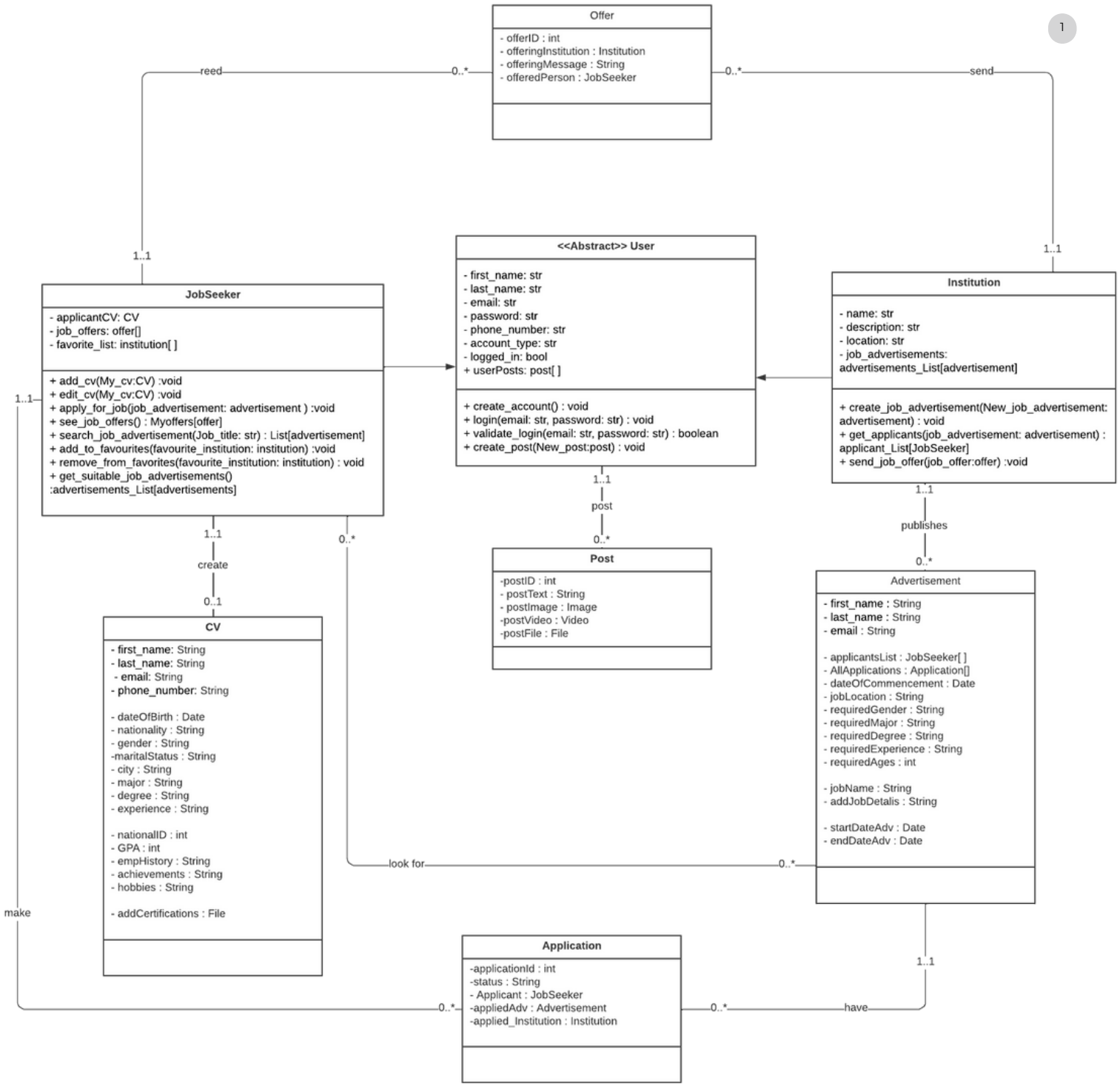


DFD diagram

level 2

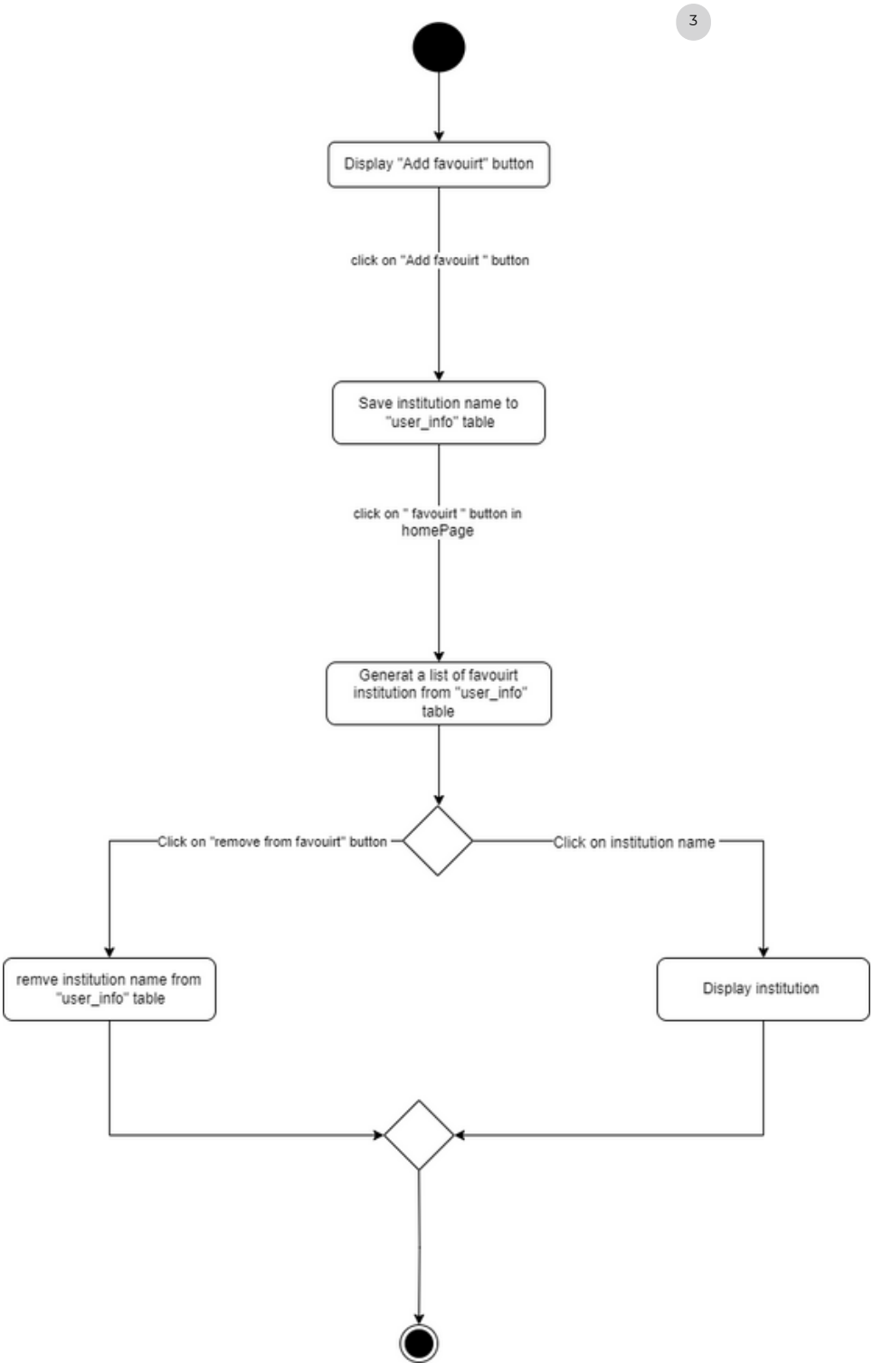


Class diagram



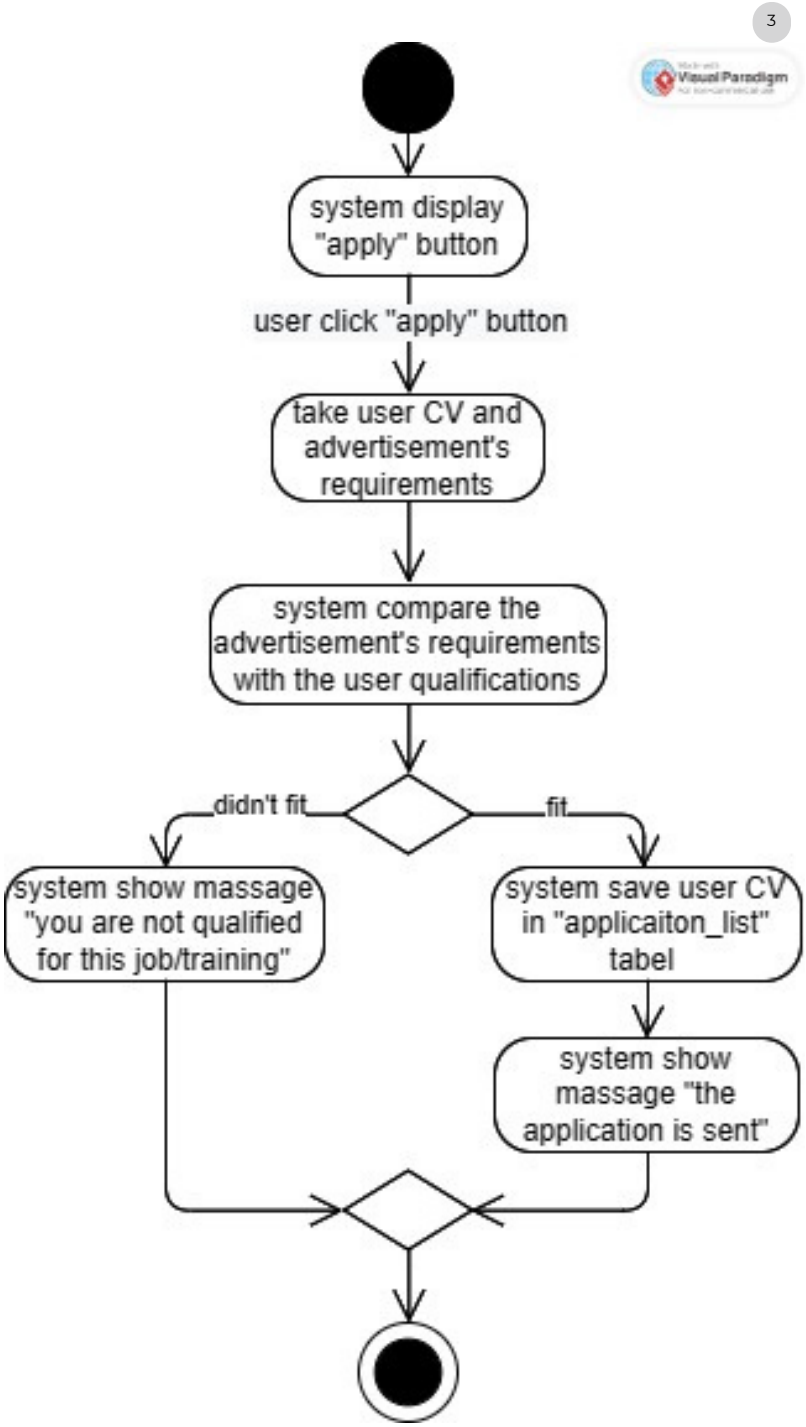
Activity diagram

"Add to favorite"



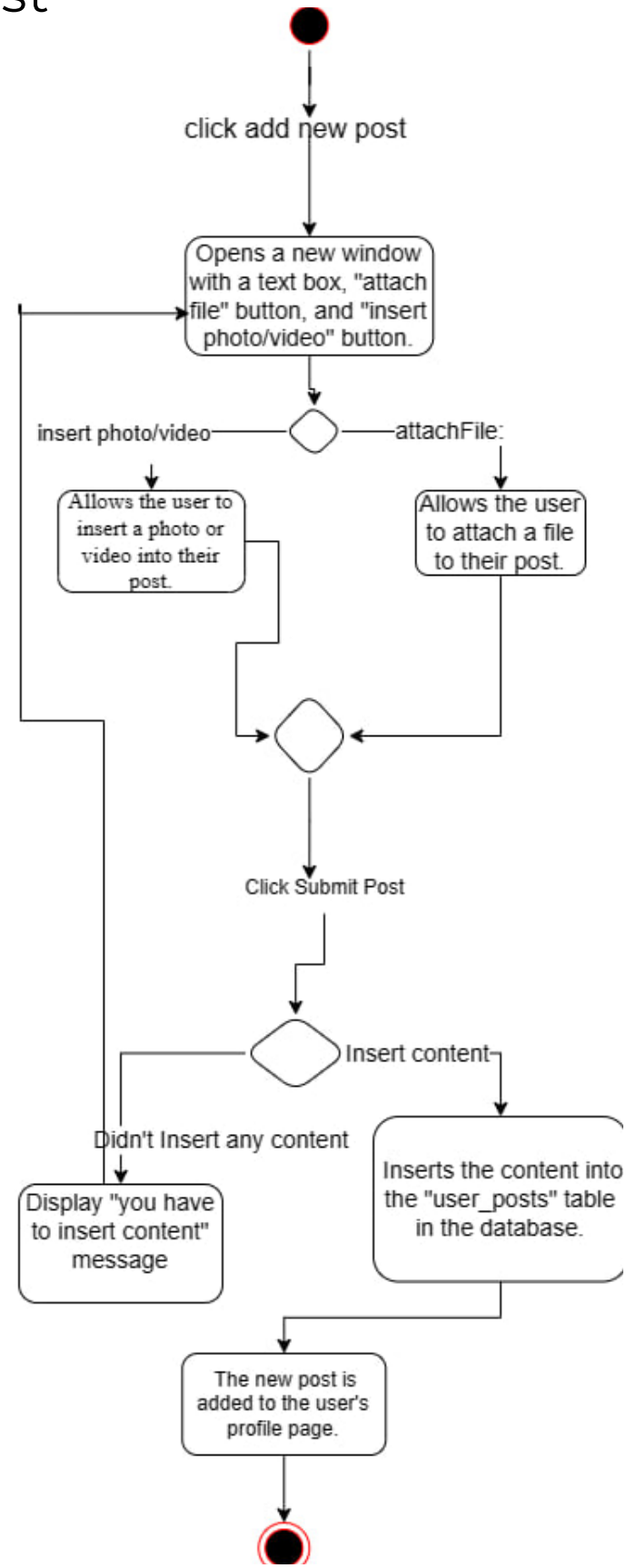
Activity diagram

"Apply for a job"



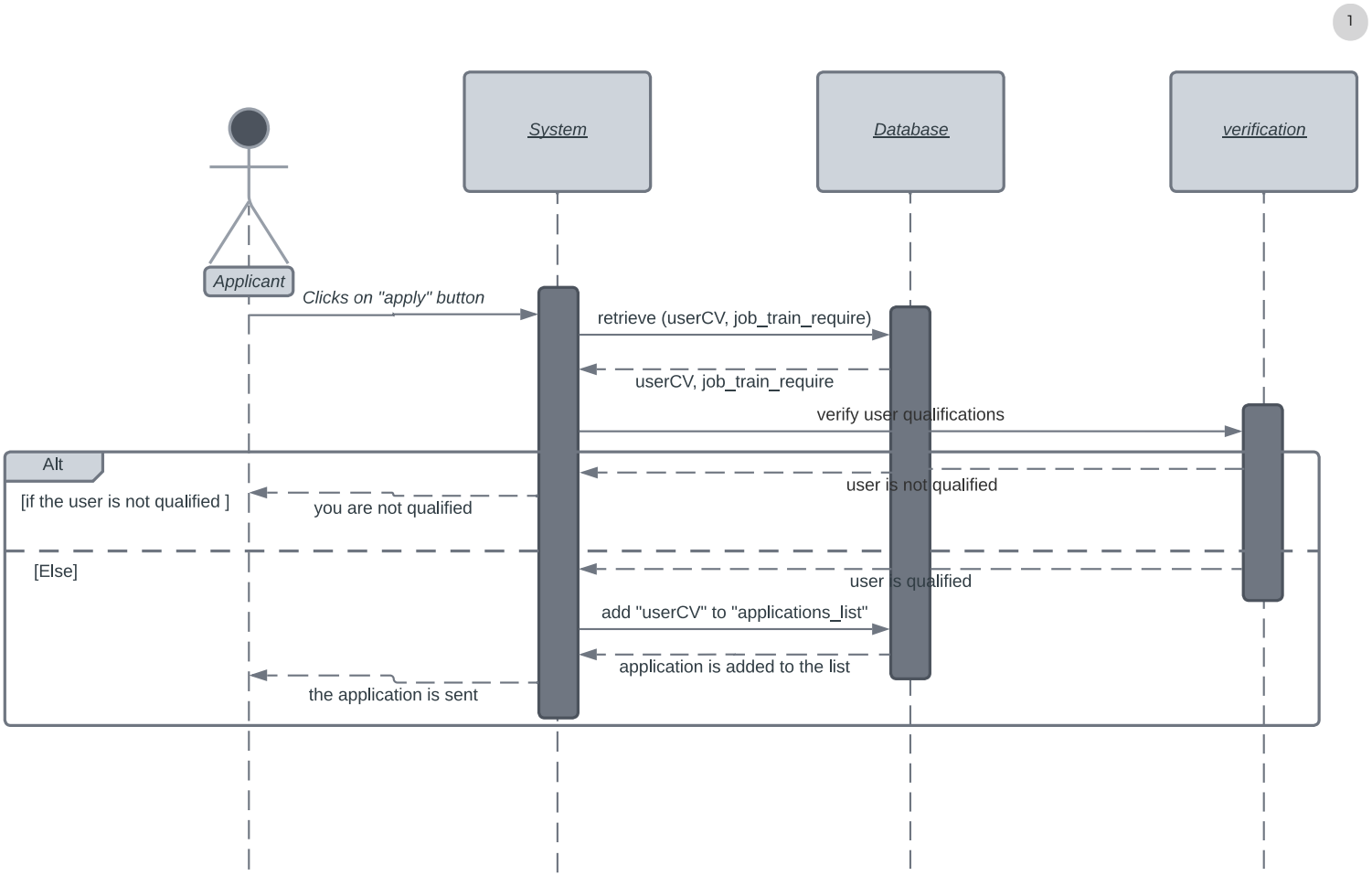
Activity diagram

"Add post"



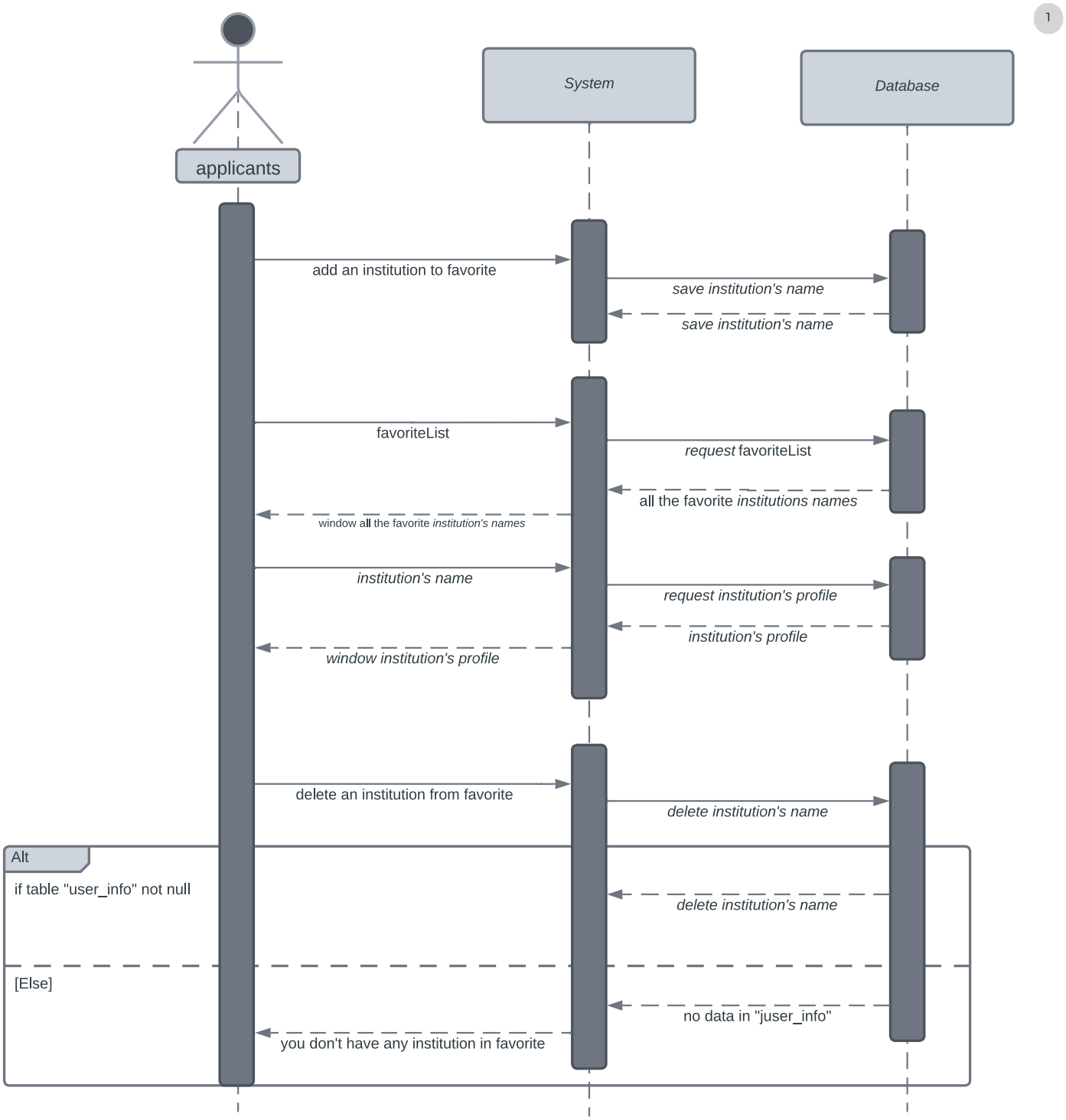
Sequence diagram

" Apply for a job"



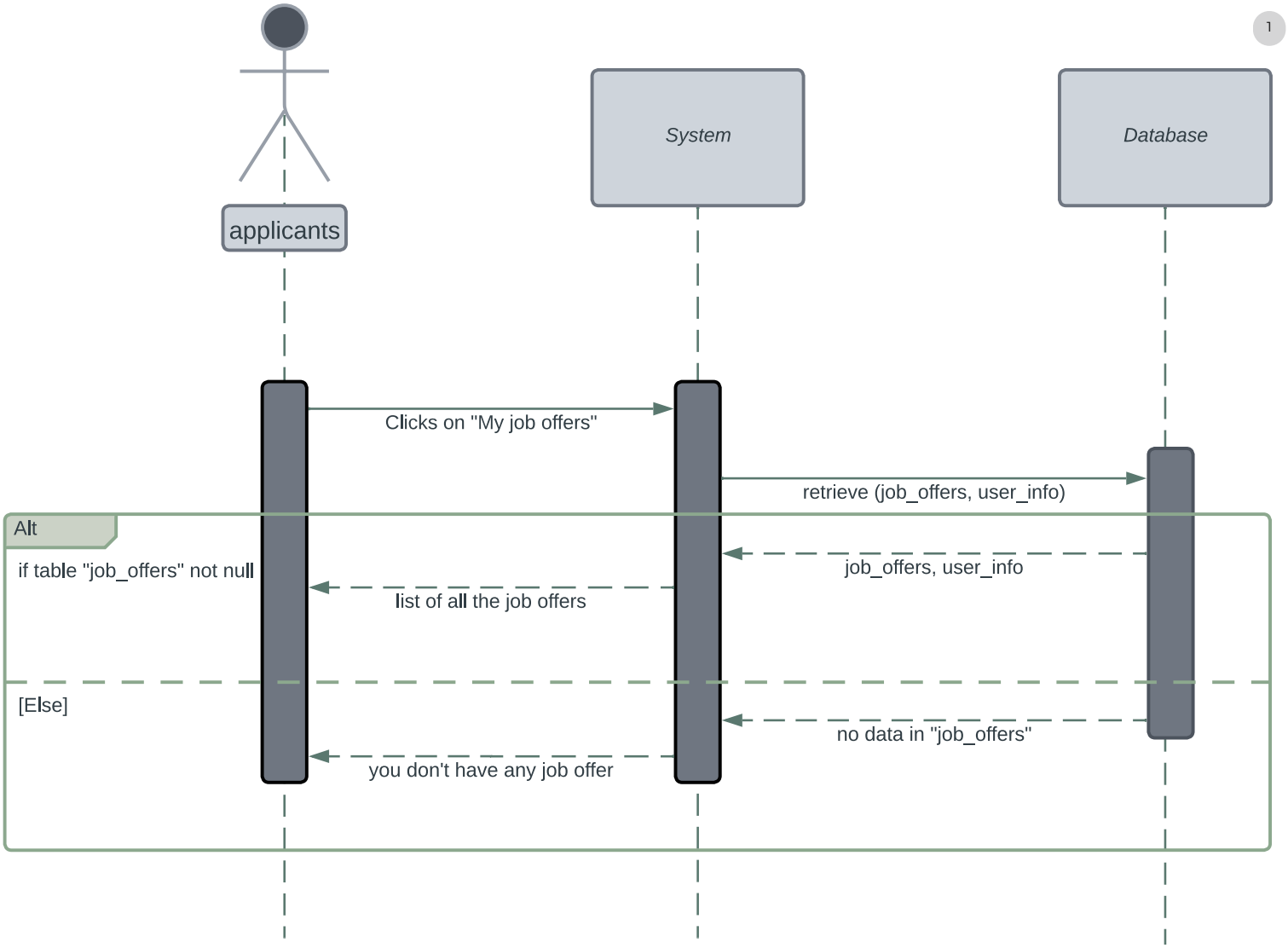
Sequence diagram

" Favourite list"



Sequence diagram

" See job offers"



Chapter 2

Design

Architecture

Architecture:

Model-View-Controller

Reason for selecting this architecture:

our software is based on action and reaction from both the user and the system, with the goal of acting as a bridge to connect applicants (the primary users of the application) to institutions, allowing users to create detailed profiles, including personal information and skill sets, and institutions to post job advertisements and internship opportunities. This consolidated platform will make contacts between applicants and institutions more efficient and effective, streamlining the process and enhancing overall satisfaction.

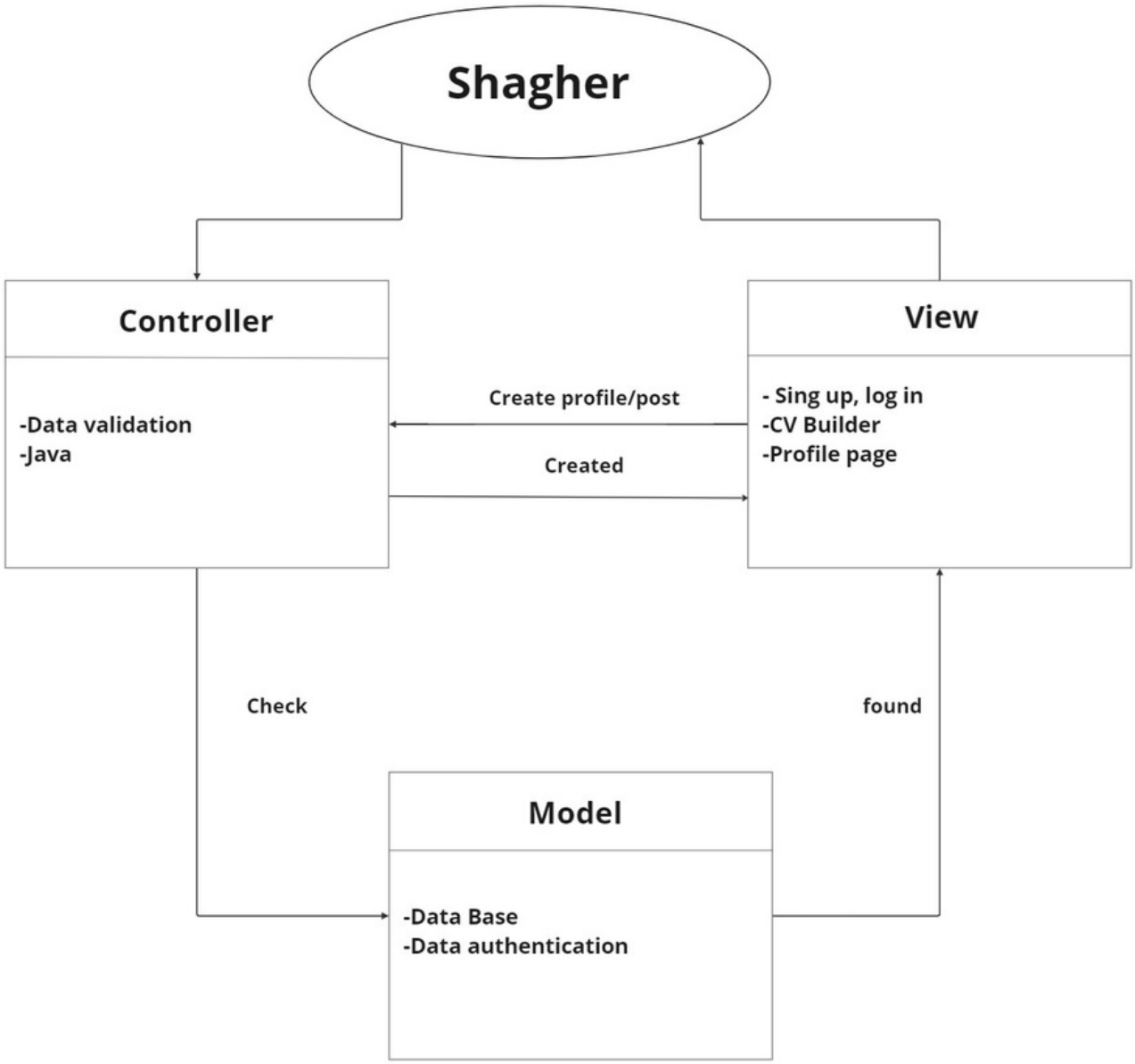
Hardware Architecture:

CLIENT-SERVER PATTERN

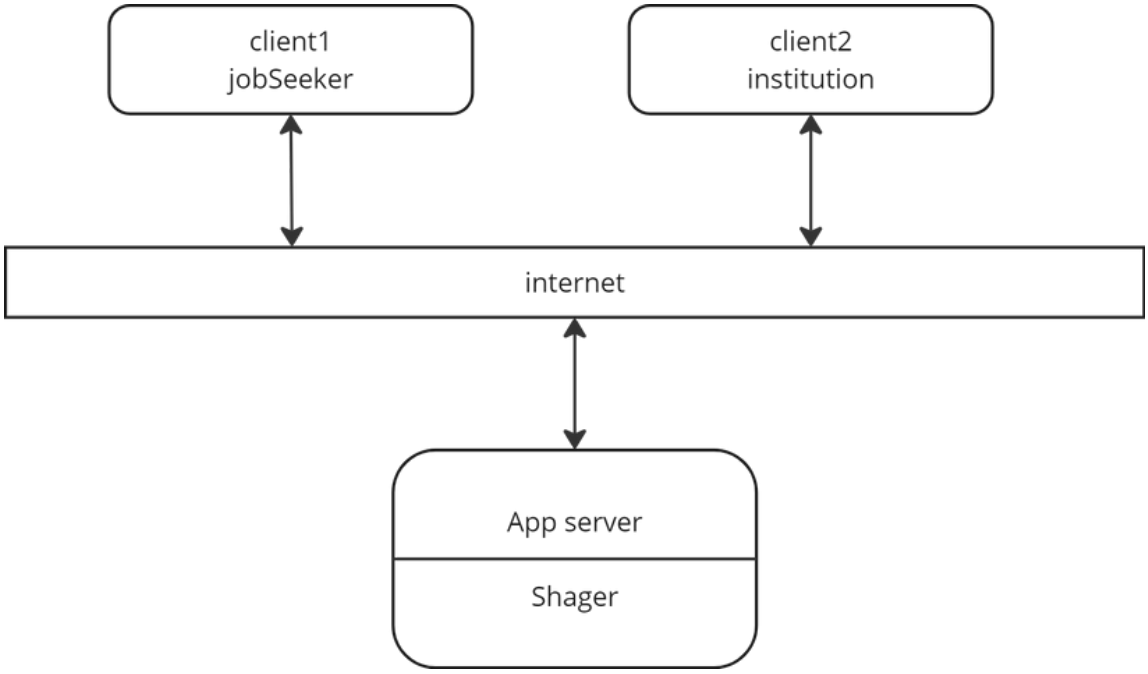
Reason for selecting this architecture:

We chose this type of architecture because we believe it is better suited for our project idea given that we have two clients (institutions and job seekers). Each client is an end user and has access to the service through the Internet.

software Architecture .....

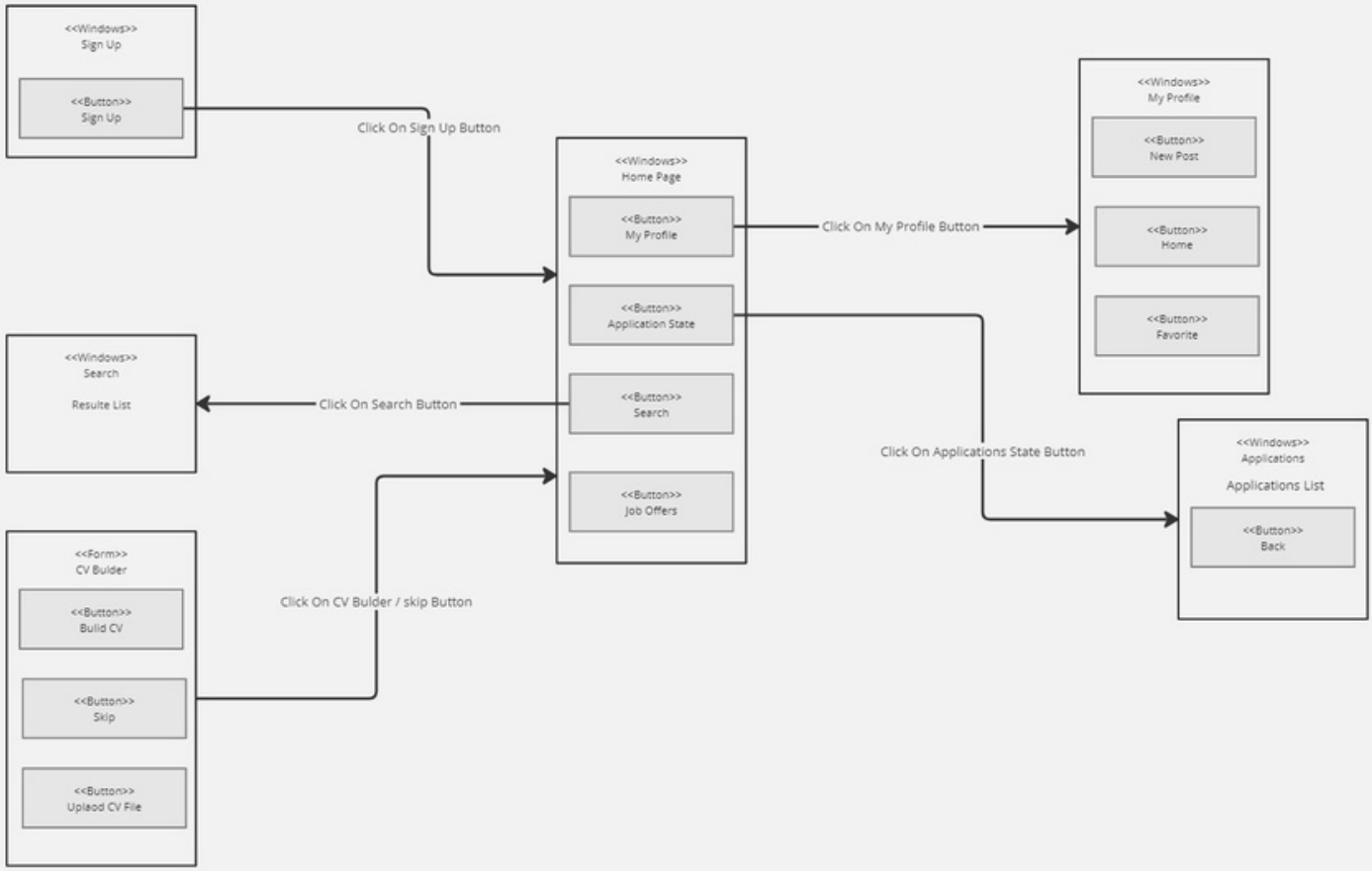


Hardware Architecture.....



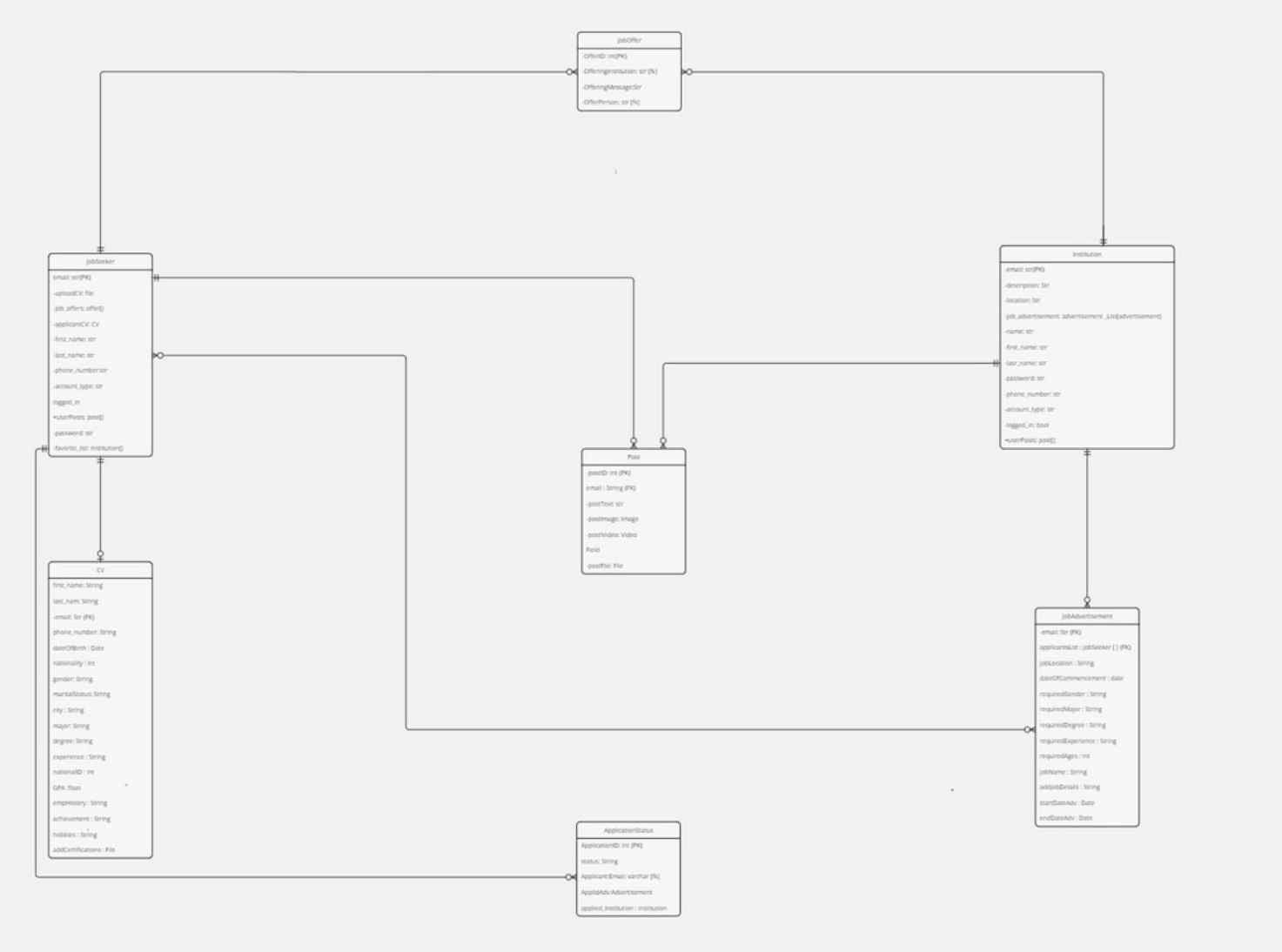
Interface Design (WND) •••••

Click here



Data Storage Design (ERDs) .....

Click here



Data Storage Design (ERDs)

Optimization Technique:

Denormalization

Reason for choosing this technique:

An optimization strategy in which we add redundant data to one or more tables is the process of adding precomputed redundant data to an otherwise normalized relational database to increase read performance. In a relational database, this can help us avoid costly joins.

Denormalized Tables:

User_info

1- When the user, whether it's a job seeker or an institution, fills in their information for the first time, the information will be stored in this table.

2- When the user performs a login and already has an account, the information will be verified by retrieving data from this table.

3- When the job seeker fills in their CV, some fields such as email, phone number, and name will be automatically filled from this table.

4- When the user marks a specific institution as a favorite, it is stored in this table. Similarly, when they perform deletion, it will be removed from this table.

Data Storage Design (ERDs)

5- When the job seeker clicks on a specific institution, the displayed information will be retrieved from this table.

6- The contact emails will be stored in this table.

userCV

1- When the user creates a CV for the first time or uploads it from a previous file, the CV data will be stored in this table.

2- When the user clicks on a specific job application, the qualifications in the CV will be compared with the requirements of the application.

3- The institution can display the CV information of the job seeker by retrieving the data from this table.

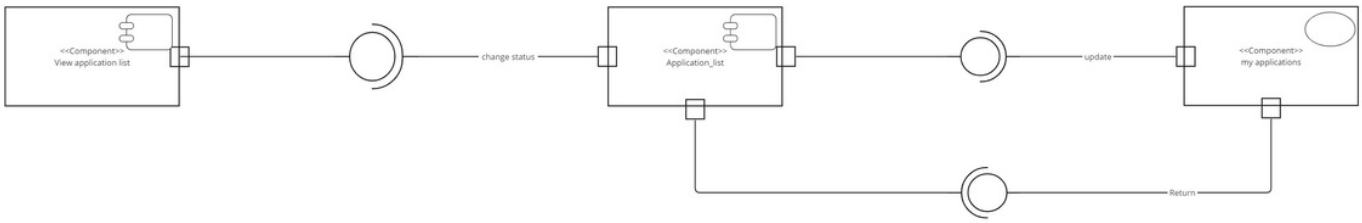
User_post

1- The user can add a new post, and it will be stored in this table.

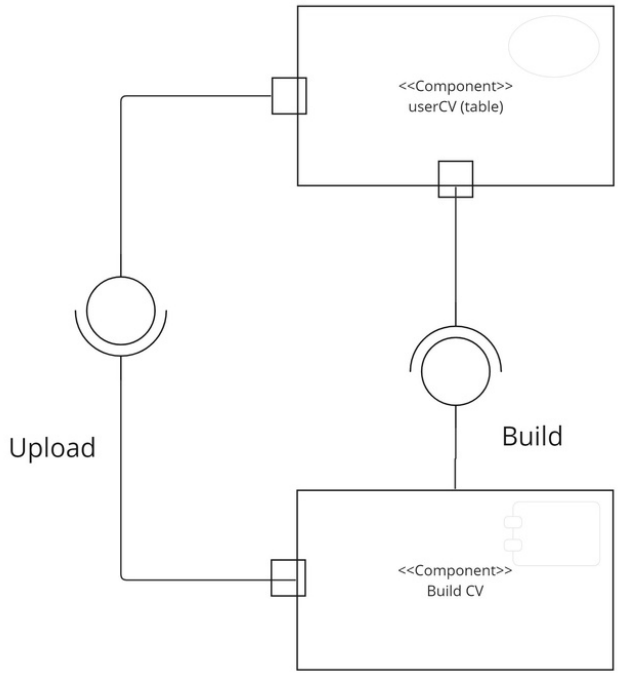
2- If the job seeker displays company information, the posts from this table will also be displayed.

Component Design

1- Application component for institution (Layan)

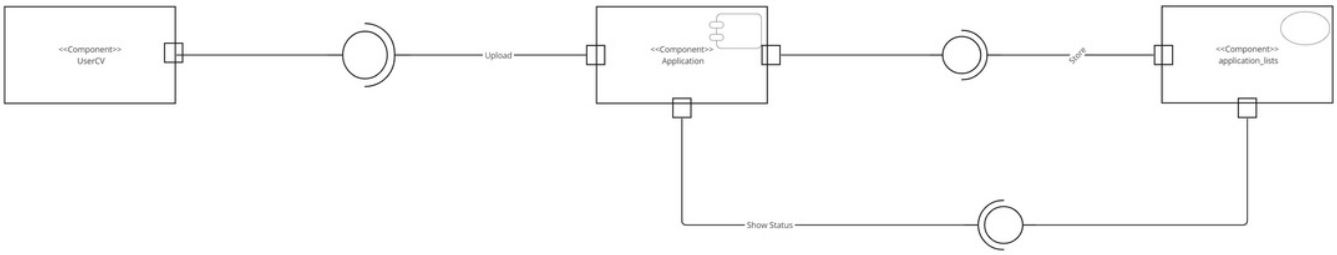


2-Build CV Component diagram (Thana'a)

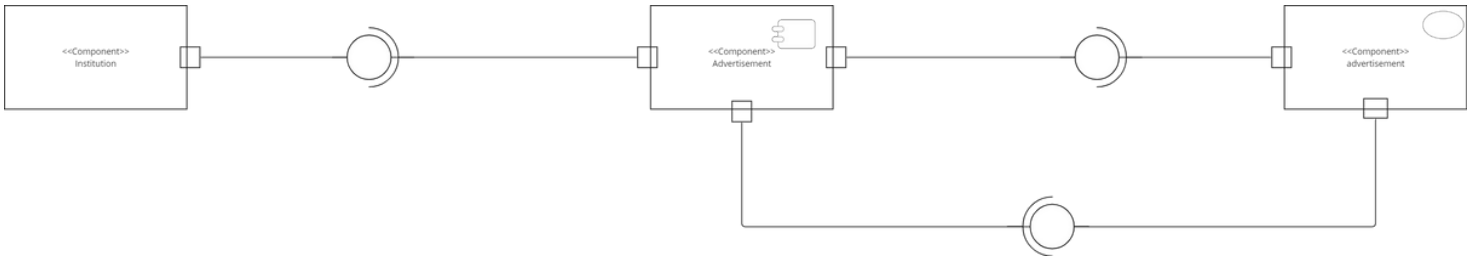


Component Design

3-Application component (Raghad)

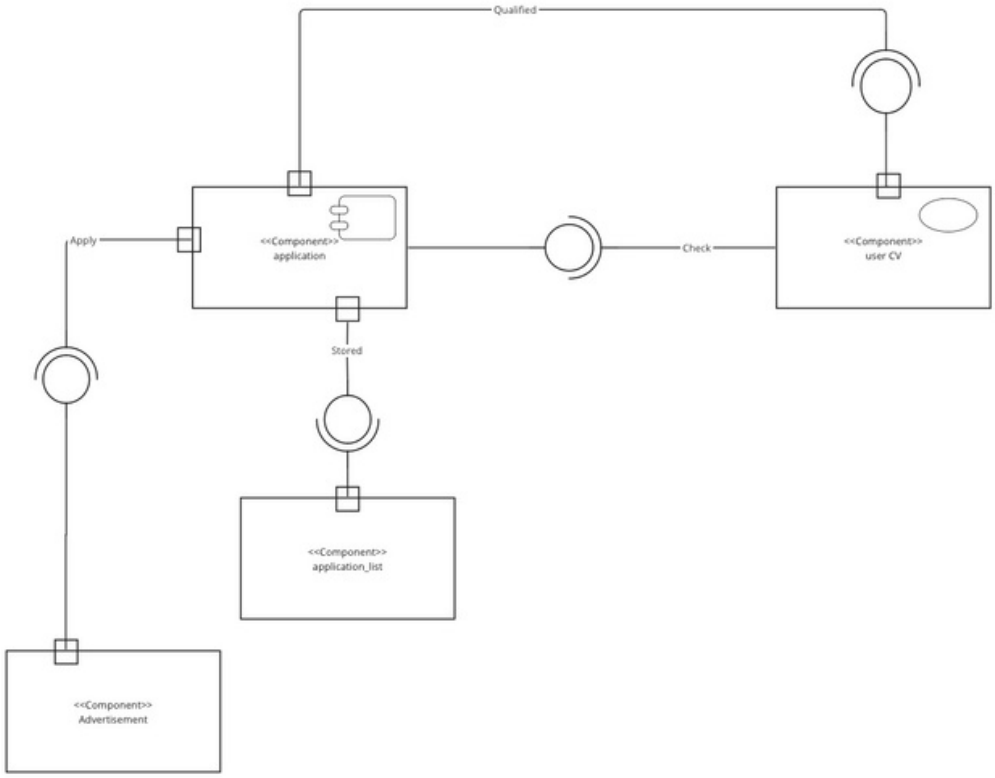


4-Advertisement component diagram from institution pov (Amjad)



Component Design

5- Applying for a job component diagram (Razan)

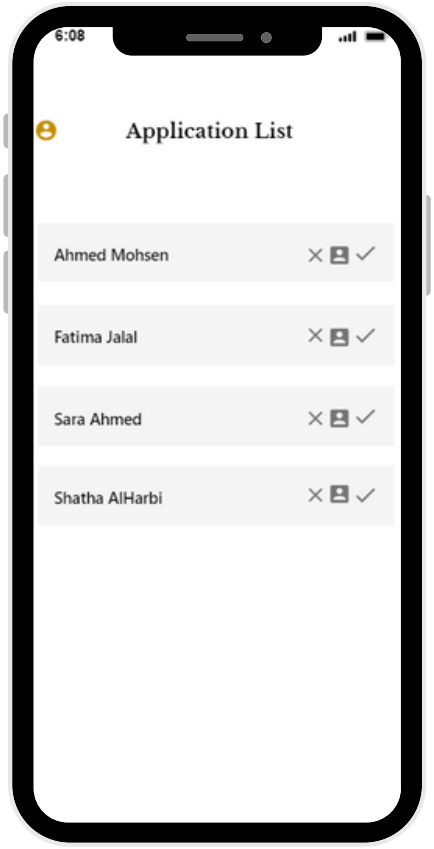


Chapter 3

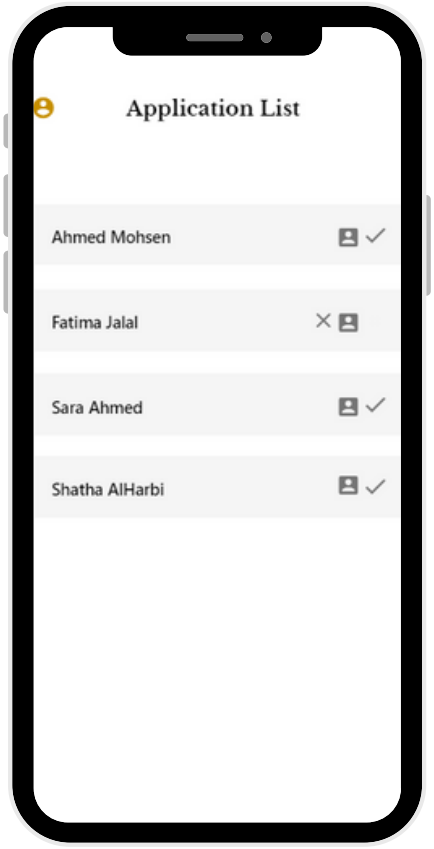
Implementation

implementation

1-application component POV institution acceptance/rejection (Raghad)



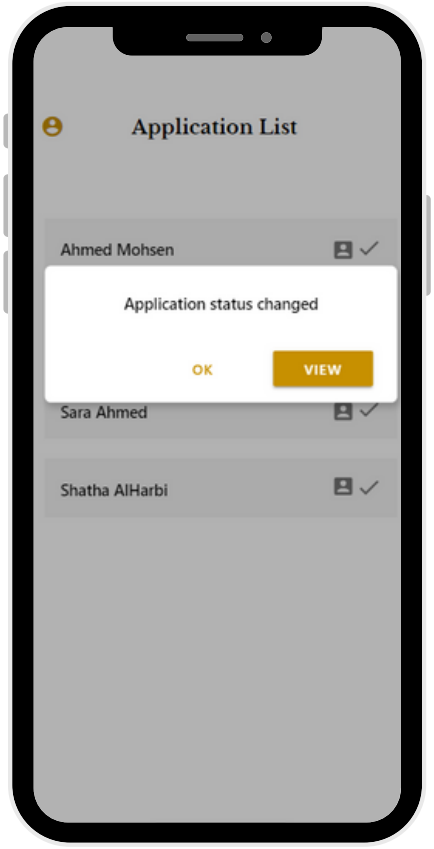
The company can see the names of applicants and their CVs



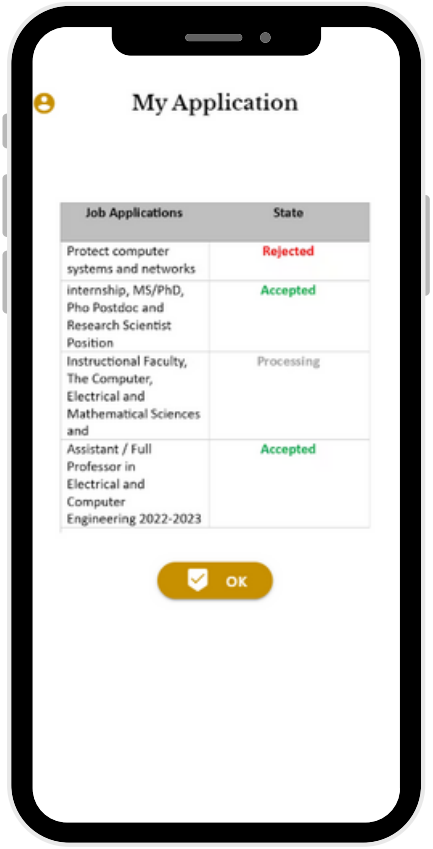
They can click the button to accept, reject, or view the CV

implementation

1-application component POV institution acceptance/rejection (Raghad)



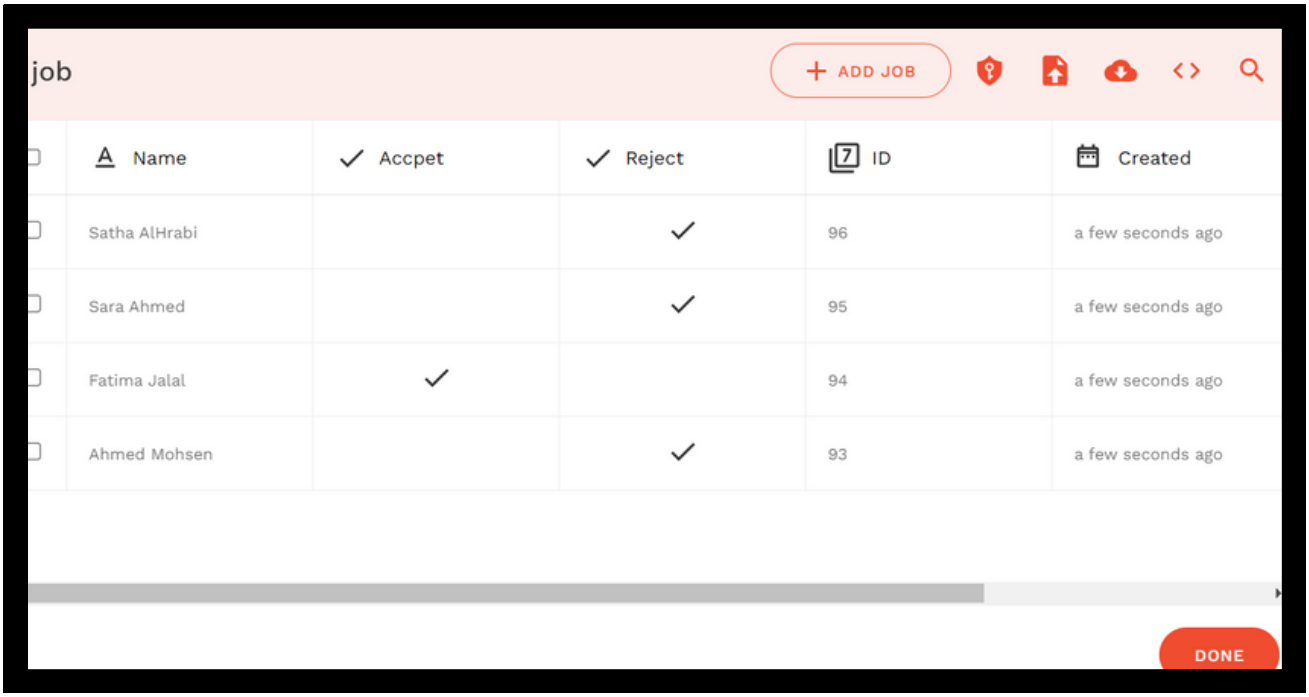
When the institution approves or rejects, a notification will arrive to the applicant that the status of the application has changed



f the applicant clicks on View, all applications will be displayed

implementation

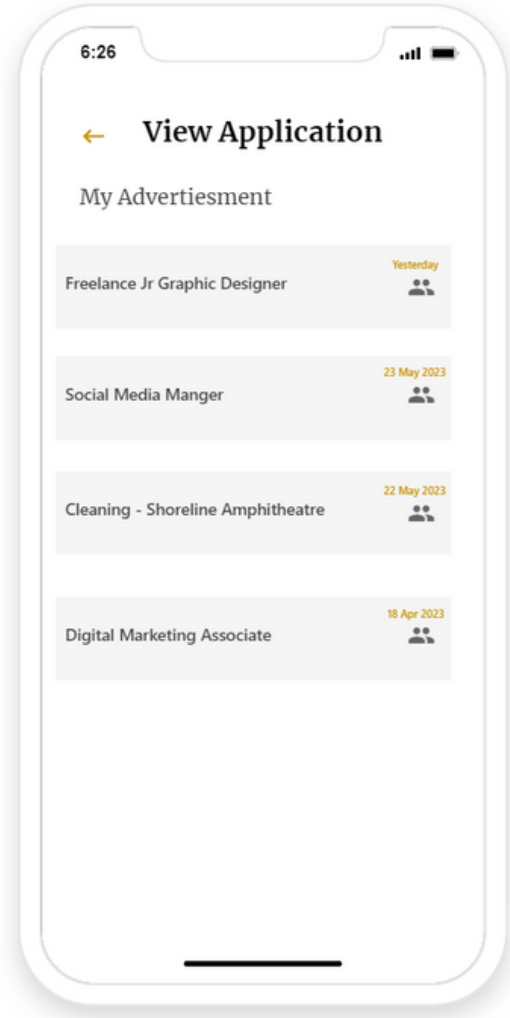
1-application component POV institution acceptance/rejection (Raghad)



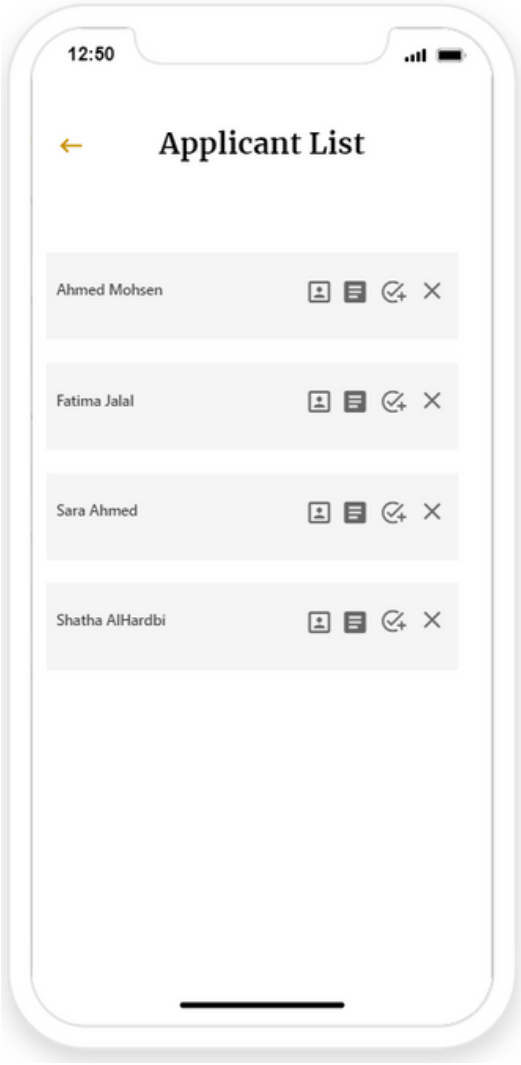
Thus, the database will display the name of the applicant and whether he is rejected or accepted

Implementation

2- Application component for institution (Layan)



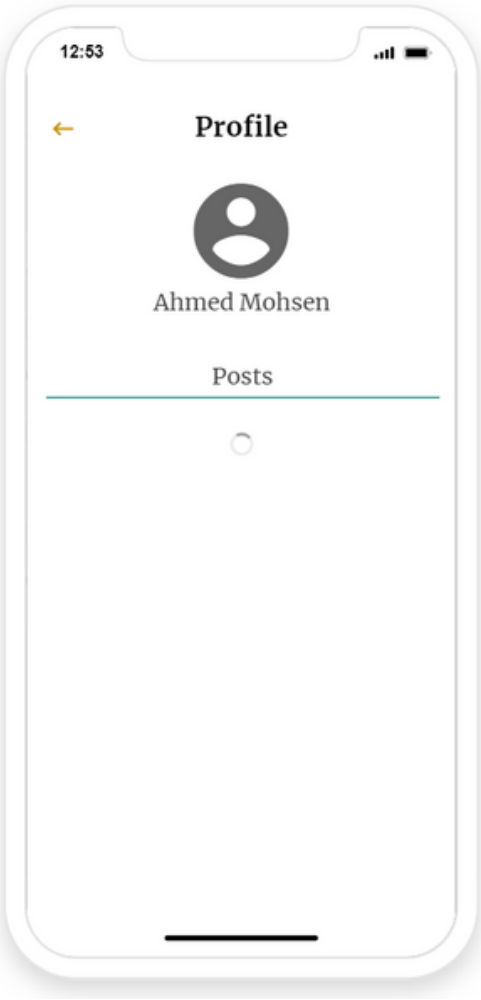
The company can see the applicants for each Advertiesment.



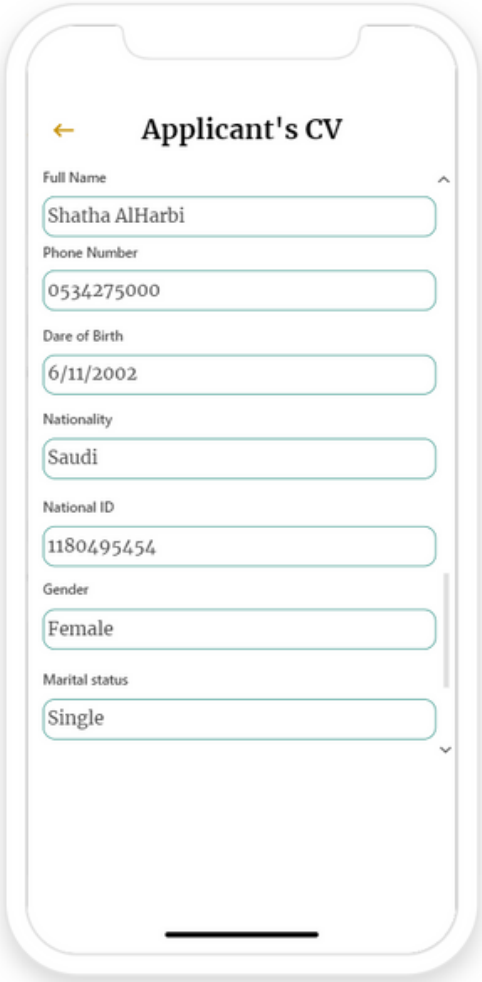
The company can see the names of applicants.

Implementation

2- Application component for institution (Layan)



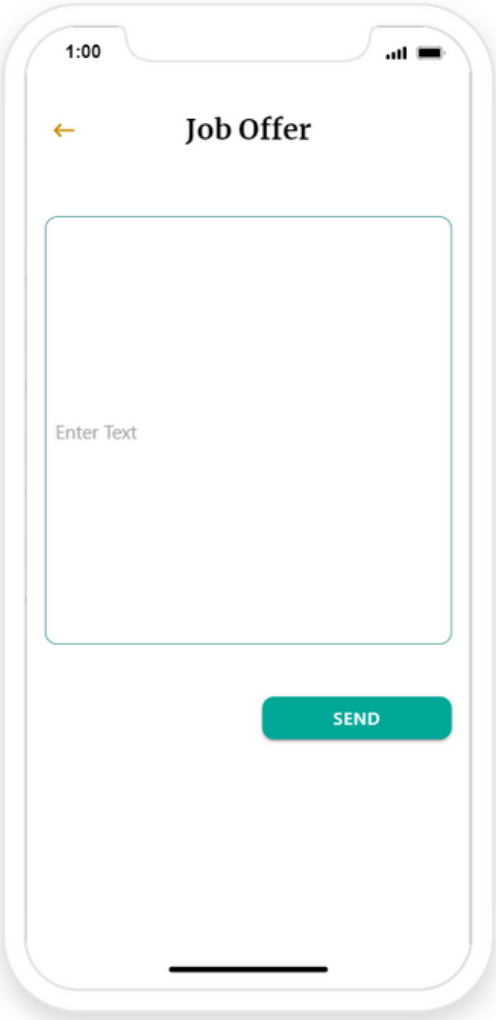
The company can see the applicant's profile.



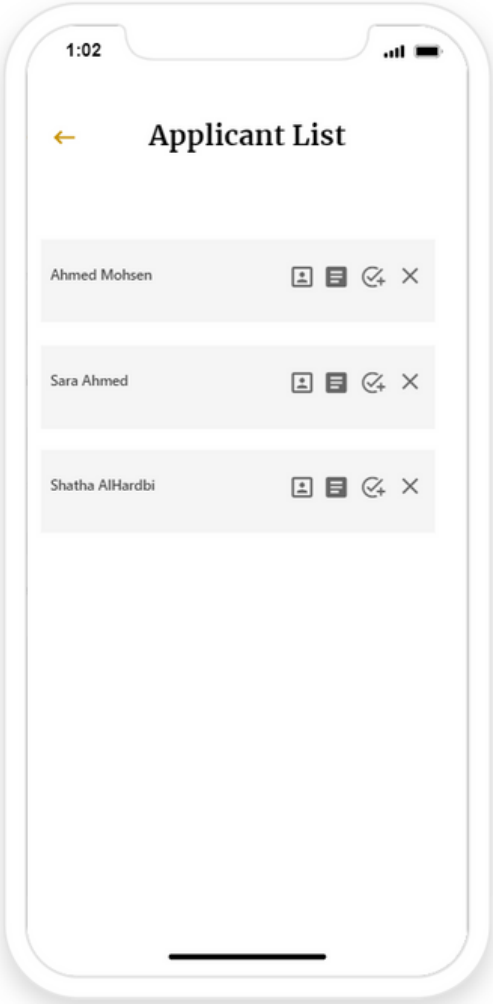
The company can see the applicant's CV.

Implementation

2- Application component for institution (Layan)



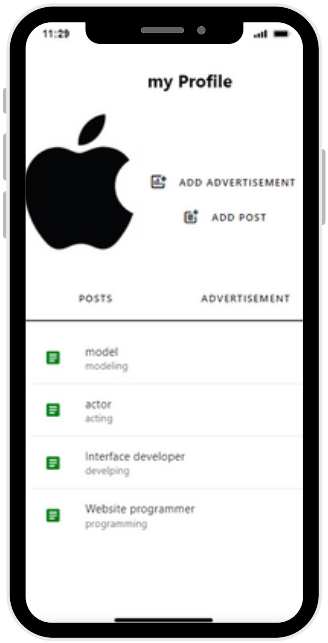
The company can accept the applicant's application and send a job offer.

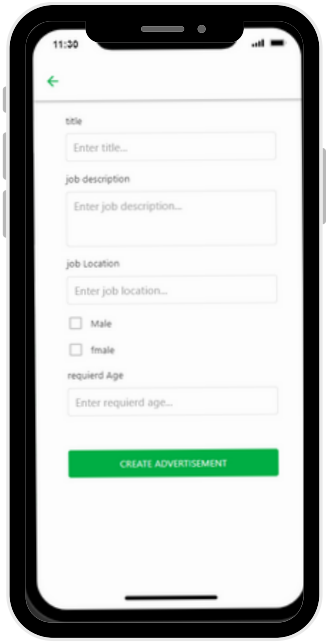


The company can decline the applicant's application.

Implementation

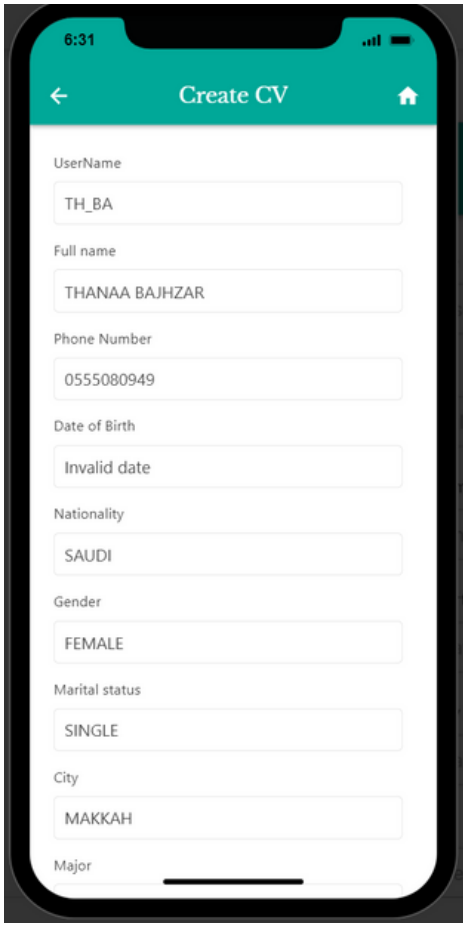
3-Advertisement component diagram from institution pov (Amjad)

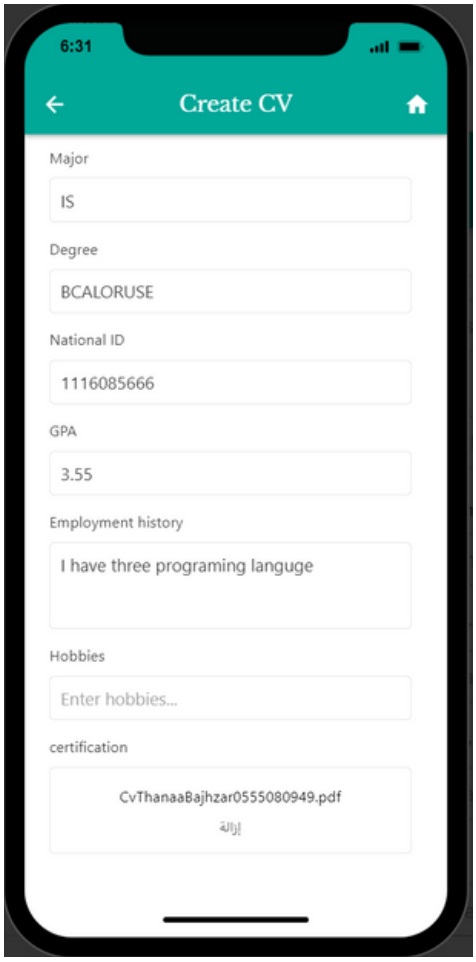




Implementation

4- Testing for Application component FORM (Thana'a)





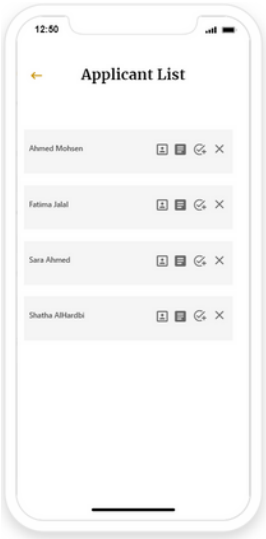
The job seeker can fill out the CV form.

Chapter 4

Testing

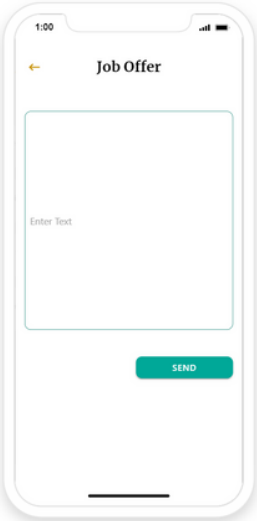
Testing

1- Testing for Application component for institution (Layan)

Sl.No	Test Case	Input	Expected output	Actual output	Remarks
1	Checking Applicants for my Advertiesment	applications.	Redirecting to the Applicants list		Pass
2	Checking the profile of one of the Applicants	profile info.	Redirecting to the Applicant's profile.		Pass

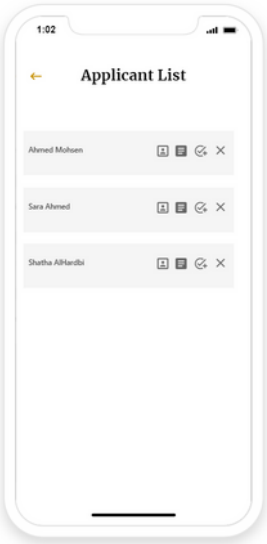
Testing

1- Testing for Application component for institution (Layan)

Sl.No	Test Case	Input	Expected output	Actual output	Remarks
3	Checking Applicant's CV	Applicant CV.	Redirecting to the Applicant's CV		Pass
4	Accepting Applicant's application	Accept.	Redirecting to the Job offer page.		Pass

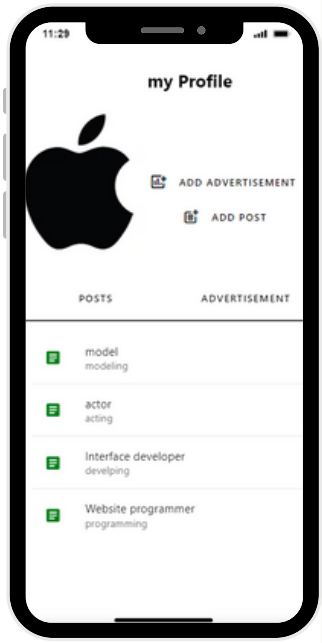
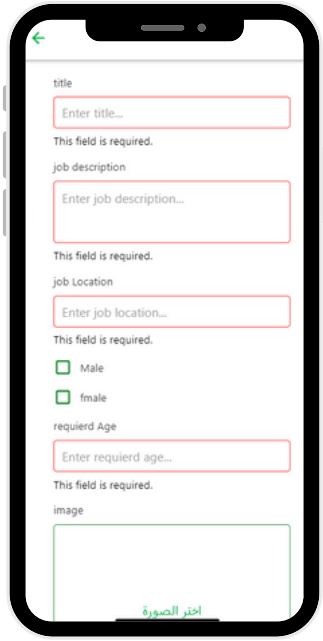
Testing

1- Testing for Application component for institution (Layan)

Sl.No	Test Case	Input	Expected output	Actual output	Remarks
5	Declining Applicant's application	Clicking on the decline button.	Redirecting to the Applicant's list after updating it		Pass

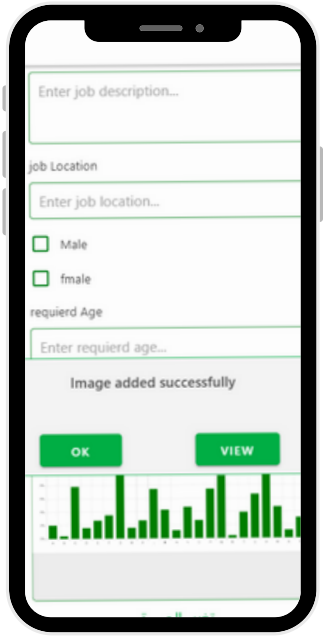
Testing

2-Advertisement component diagram from institution pov (Amjad)

Sl.No	Test Case	Input	Expected output	Actual output	Rem arks
1	Add advertisement	Advertisement informations.	Add to database and show in list		Pass
2	Add advertisement without the required fildes	empty fildes	Show empty fields in red boxes and "This field is required" error message		Pass

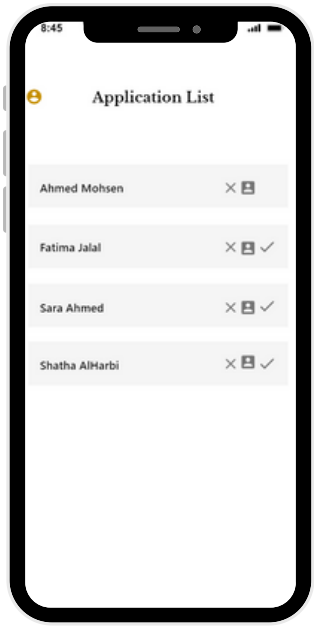
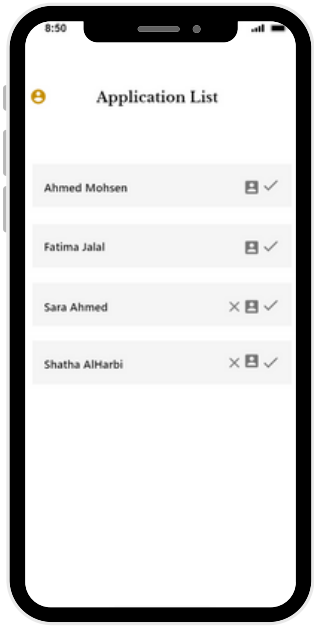
Testing

2-Advertisement component diagram from institution pov (Amjad)

Sl.No	Test Case	Input	Expected output	Actual output	Remarks
3	Add image to the Advertisement.	image	upload the image and save it in the database		Pass

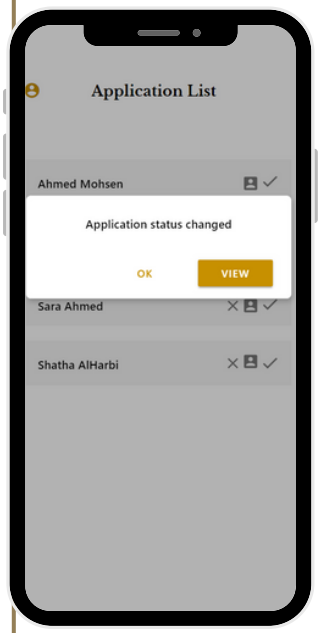
Testing

3-application component POV institution acceptance/rejection (Raghad)

Sl.No	Test Case	Input	Expected output	Actual output	Remarks
1	jobseeker acceptance	click on button “accept”	put check mark at “Acceptable” in database in front of jobseeker name		Pass
2	jobseeker rejectence	click on button “reject”	put check mark at “Rejectance” in database in front of jobseeker name		Pass

Testing

3-application component POV institution acceptance/rejection (Raghad)

Sl.No	Test Case	Input	Expected output	Actual output	Remarks
3	send notification	text	click on view button to see the application page		Pass

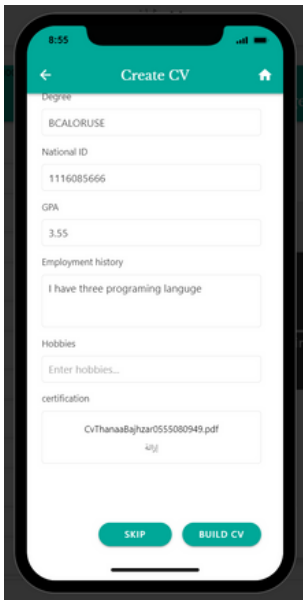
Testing

4- Testing for Application component FORM (Thana'a)

Sl.No	Test Case	Input	Expected output	Actual output	Remarks
1	HOME button	Click Home page.	The home page appears		Pass
2	Display a window for entering the date of birth.	Date of birth	The window for entering the date of birth is appears.		Pass

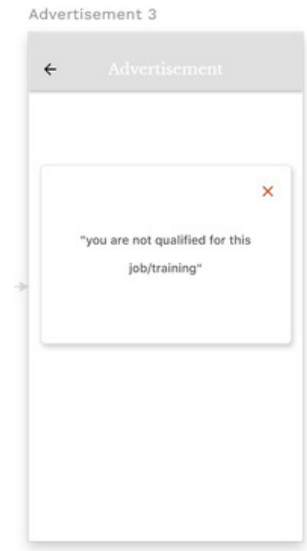
Testing

4- Testing for Application component FORM (Thana'a)

Sl.No	Test Case	Input	Expected output	Actual output	Remarks
3	Upload file.	CV File	can upload		Pass

Testing

5- Applying for a job component diagram (Razan)

Sl.No	Test Case	Input	Expected output	Actual output	Remarks
1	test case for valid entry	correct verification code	verification page		Pass
2	the case for invalid entry	wrong verification code	verification page		Pass

Chapter 5

Evolution

Evoluation

1-application component POV institution acceptance/rejection (Raghad)

The maintenance technique is: Refactoring
The issue we want to solve: the presence of ***duplicate code*** in the application component of the software system. This duplication occurs in various lines, including retrieving job/training advertisement names, applicant names, displaying applicant CVs, providing accept/reject buttons, saving application status, and retrieving applicant profile information.
How we're going to Solve it: To address this problem effectively, it is recommended to refactor the code by creating reusable methods or functions.
Examples of such functions are:
"retrieveApplicantNames,"
"displayApplicantCV,"
"updateApplicationStatus," and "retrieveApplicantProfile." By implementing this solution, the codebase will become more maintainable, readable, and free of unnecessary repetition.

2- Posts in user's profile (Layan)

The maintenance technique is: Refactoring
The issue we want to solve: The redundancy that occurs When the same collection of data items (fields in classes, arguments in methods), appears repeatedly throughout our software.
How we're going to Solve it: By adding all-encompassing methods or functions in the many classes that deal with or interface with the post component from either the organization or the application users.
Examples of such functions are:
"retrieveOrganizationPost" and "retrieveUserPost"

Evoluation

3-Advertisement component diagram from institution pov (Amjad)

The maintenance technique is: Refactoring
The issue we want to solve: The problem we may encounter is code duplication in the advertising component of the software system. It will be copied in different lines, including creating an ad, adding an ad, and showing it in the list.
How we're going to Solve it: To effectively address this issue, it is recommended to refactor the code by creating reusable methods or functions.
Examples of such functions are:
"Create an ad"
"Add an ad"
"Show ads list"

4- Generalized methods to reduce delay in search in advertainment. (Thana'a)

The maintenance technique is: Refactoring
The issue we want to solve: Data clumping can have several negative effects on an application:

- Reduced readability: When related data is scattered throughout the code, it can make the code harder to read and understand.
- Code duplication: Developers may end up duplicating data in different parts of the code, leading to potential inconsistencies and bugs.

How we're going to Solve it: How we're going to Solve it: Generalized methods to reduce delay in search in advertainment.
Examples of such functions are: search in Database.

Evoluation

5- Applying for a job component diagram (Razan)

The maintenance technique is: Refactoring
The issue we want to solve: repeating the programming processes to match the CV data and the advertisement data, and more than one line will be copied.
How we're going to Solve it: To solve this problem effectively, it is to include a method that performs this process for reuse.