swingroup04

Welcome to your team space!

- We've added some suggestions and placeholders. Everything is customizable.
- Get started with page templates:
  - Project plan
  - Meeting notes [19-09-23]
  - https://group04swin.atlassian.net/wiki/spaces/group04/pages/360758 Can't find link
- Check out Get the most out of your team space UNDEFINED for more tips.

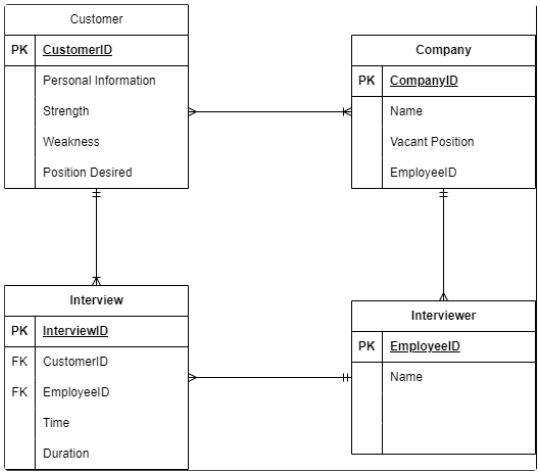
Project Proposal

COS20031 - Class 2 - Group 4

Entity Relationship Diagram

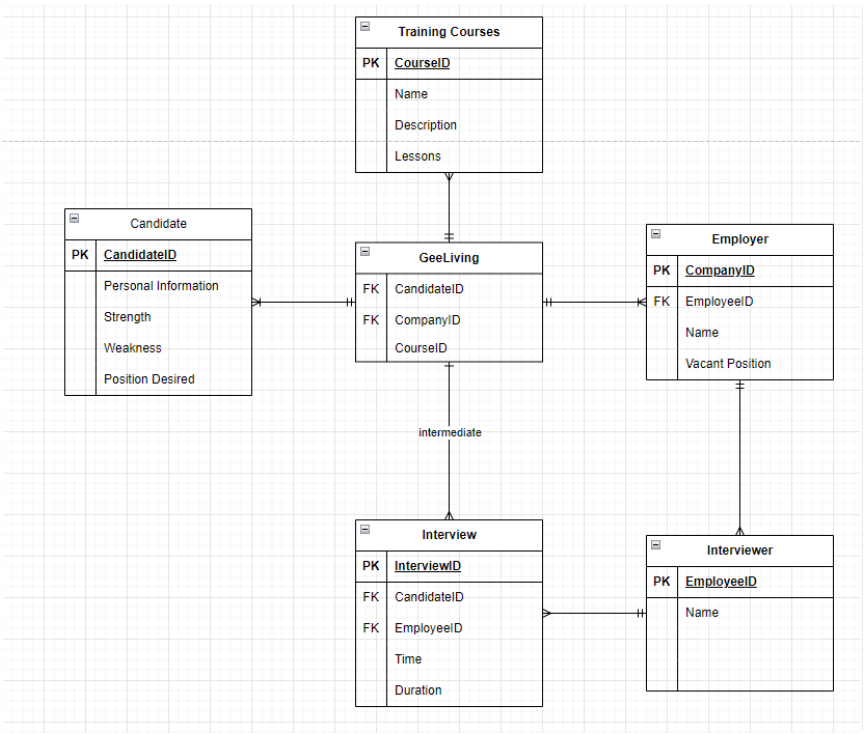
This page exists to show every version of the Entity Relationship Diagram that we made.

Initial Entity Relationship (ER) Diagram (02/10/2023)



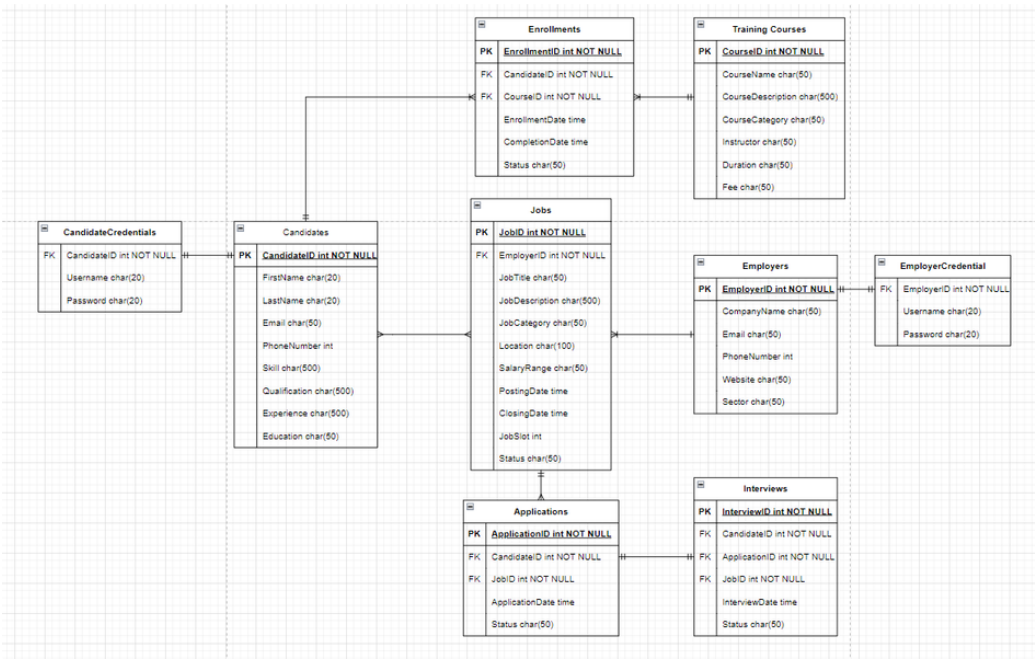
The original Entity Relationship Diagram that we made to basically determine the neccessary elements needed to start with the project. We determined that 4 basic elements, which includes the Customer who use the website for emplyment, the Company who will put the notice for hiring on the website, and subsequently the Interviewer who belongs to that Company and the Interview conducted by the Interviewer and the Customer.

Updated Entity Relationship (ER) Diagram (20/10/2023)



We changed what we thought to be Customer to Candidate, as the Employer can also be counted as a Customer. We extend the process by adding GeeLiving table to act as an intermediate. We also add into the diagram another extension, the Training Courses. They belongs to the GeeLiving and have their own name, description and lessons.

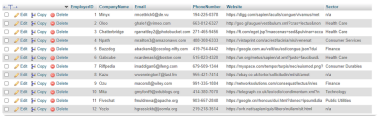
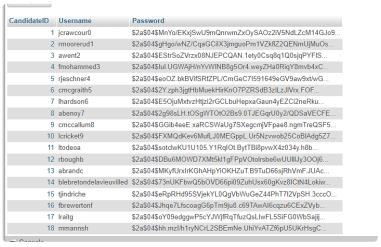
Updated Entity Relationship (ER) Diagram (14/11/2023)



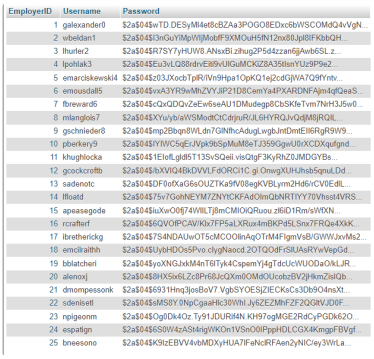
The final Entity Diagram that we have decided upon. In this diagram we have decided to completely remove the GeeLiving table as it is not really suitable. Therefore, the common between Candidates and Employers will be Jobs - the one that Candidate wants to apply to, and Employers want people from. We have also extended the application for Interviews, as the Employers might decide a candidate not suitable for the job even before interview. We have also added an additional Credientals for both Candidate and Employer as their account log in details. Finally, some details have been added to Training Courses, as well as Enrollments table to record the subject for Enrollment and the candidate who enroll it.

Physical Database - Create Table (Week 6)

For the physical database, our group will be utilizing PHPMyAdmin to create table, from there we will update the relationship between each of them

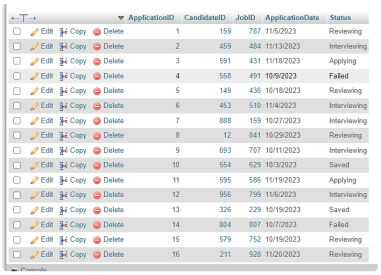
Table Name	Image	Query
Candidates		```sql CREATE TABLE `candidates` (`CandidateID` int(50) NOT NULL, `FirstName` varchar(50) DEFAULT NULL, `LastName` varchar(50) DEFAULT NULL, `Email` varchar(50) DEFAULT NULL, `PhoneNumber` varchar(50) DEFAULT NULL, `Skill` varchar(500) DEFAULT NULL, `Qualification` varchar(500) DEFAULT NULL, `Experience` varchar(500) DEFAULT NULL, `Education` varchar(500) DEFAULT NULL) ```
Employers		```sql CREATE TABLE `employers` (`EmployerID` int(10) NOT NULL DEFAULT '0', `CompanyName` varchar(50) DEFAULT NULL, `Email` varchar(50) DEFAULT NULL, `PhoneNumber` varchar(50) DEFAULT NULL, `Website` varchar(50) DEFAULT NULL, `Sector` varchar(50) DEFAULT NULL) ```
Jobs		```sql CREATE TABLE `jobs` (`JobID` int(10) NOT NULL, `EmployerID` int(10) NOT NULL, `JobTitle` varchar(50) DEFAULT NULL, `JobDescription` varchar(500) DEFAULT NULL, `JobCategory` varchar(50) DEFAULT NULL, `Location` varchar(100) DEFAULT NULL, `SalaryRange` varchar(50) DEFAULT NULL, `PostingDate` varchar(50) DEFAULT NULL, `ClosingDate` varchar(50) DEFAULT NULL, `JobSlot` int(10) NOT NULL, `Status` varchar(50) DEFAULT NULL) ```
CandidateCredentials		```sql CREATE TABLE `candidatecredentials` (`CandidateID` int(50) NOT NULL, `Username` varchar(50) DEFAULT NULL, `Password` varchar(100) DEFAULT NULL) ```

EmployerCredentials



```
CREATE TABLE `employercredentials` (
  `EmployerID` int(10) DEFAULT NULL,
  `Username` varchar(50) DEFAULT NULL,
  `Password` varchar(100) DEFAULT NULL
)
```

Applications

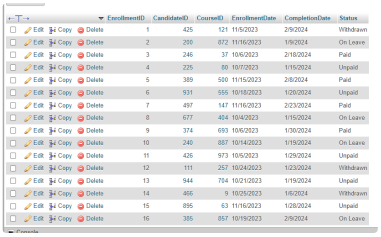


```
CREATE TABLE `applications` (
  `ApplicationID` int(10) NOT NULL,
  `CandidateID` int(10) NOT NULL,
  `JobID` int(10) NOT NULL,
  `ApplicationDate` varchar(50) DEFAULT NULL,
  `Status` varchar(50) DEFAULT NULL
)
```

TrainingCourses

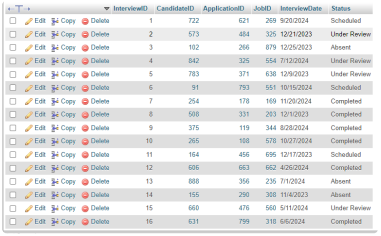
```
CREATE TABLE `trainingcourses` (
  `CourseID` int(10) NOT NULL,
  `CourseName` varchar(50) DEFAULT NULL,
  `CourseDescription` varchar(500) DEFAULT NULL,
  `CourseCategory` varchar(50) DEFAULT NULL,
  `Instructor` varchar(50) DEFAULT NULL,
  `Duration` varchar(50) DEFAULT NULL,
  `Fee` varchar(50) DEFAULT NULL
)
```

Enrollments



```
CREATE TABLE `enrollments` (
  `EnrollmentID` int(10) NOT NULL,
  `CandidateID` int(10) NOT NULL,
  `CourseID` int(10) NOT NULL,
  `EnrollmentDate` varchar(50) DEFAULT NULL,
  `CompletionDate` varchar(50) DEFAULT NULL,
  `Status` varchar(50) DEFAULT NULL
)
```

Interviews



```
CREATE TABLE `interviews` (
  `InterviewID` int(10) NOT NULL,
  `CandidateID` int(10) NOT NULL,
  `ApplicationID` int(10) NOT NULL,
  `JobID` int(10) NOT NULL,
  `InterviewDate` varchar(50) DEFAULT NULL,
  `Status` varchar(50) DEFAULT NULL
)
```

Document on Data Creation and Null Values (Week 7)

For Mock Creation to test on our database, we used Mockaroo to be the dataset for training. Following the ER Diagram that we had agreed on. We created the database based on 3NF format to keep everything in the table unique.

The data set can be download at: Dataset_Group_2.4_COS20031.zip
To decide which values can be Null, we had to consider on what must be on the data. For example:

- All the ID must NOT be Null, because each one of them is what determine on which it is, and they must not be changed. A thousand data will have a thousand ID
- Email, address, age, … many things else can be Null, because they can be added later, or that user simply don’t want to add them to provide additional security.

Use cases and SQL statements, transactions (Week 8)

Our group has meticulously worked on the use cases and user stories. For each case, we've described the user scenario, identified the necessary SQL statements, and determined if a transaction is needed. In cases where transactions were required, we've carefully formulated them. Our approach ensures our solutions are user-centric, efficient, and maintain data integrity.

1. Scenario 1: Candidate Management

Use Case/User Story	SQL Statements
As a candidate, I want to create, view, edit and update my personal profile so that the employer can have a more insightful understanding of my background.	INSERT INTO candidates (FirstName, LastName, Email, PhoneNumber, Skill, Qualification, Experience, Education) VALUES (X, X, X, X, X, X, X, X); UPDATE candidates SET FirstName = X, LastName = X, Email = X, PhoneNumber = X, Skill = X, Qualification = X, Experience = X, Education = X WHERE CandidateID = X;
As a candidate, I want to make an application for a desired job so that I can make a progress in my career.	INSERT INTO applications (CandidateID, JobID, ApplicationDate, Status) VALUES (X, X, X, X);
As a candidate, I want to view the status of my applications so that I can book an interview.	SELECT Status FROM applications WHERE CandidateID = X;
As a candidate, I want to plan, postpone, or cancel interviews so that I can manage my schedule.	UPDATE interviews SET InterviewDate = X WHERE InterviewID = X;

The `applications` and `jobs` tables must be updated by the system whenever a candidate applies for a position. To guarantee data consistency, this needs to be completed as a transaction. The `JobSlot` in the `jobs` table must be decreased if the application is accepted.

Here is our SQL Transaction:

```
BEGIN TRANSACTION;
    INSERT INTO applications (CandidateID, JobID, ApplicationDate, Status) VALUES (X, X, X, X);
    UPDATE Jobs SET JobSlot = JobSlot - 1 WHERE JobID = X;
    COMMIT;
```

2. Scenario 2: Employer Management

Use Case/User Story	SQL Statements
As a HR staff, I would like to create and update the company profile so that I can post available job listings.	INSERT INTO employers (CompanyName, Email, PhoneNumber, Website, Sector)

	VALUES (X, X, X, X, X); UPDATE employers SET CompanyName = X, Email = X, PhoneNumber = X, Website = X, Sector = X WHERE EmployerID = X;
As a HR staff, I would like to post, update, or remove job listings so that candidates can look for a position.	INSERT INTO jobs (EmployerID, JobTitle, JobDescription, JobCategory, Location, SalaryRange, PostingDate, ClosingDate, JobSlot, Status) VALUES (X, X, X, X, X, X, X, X, X, X);
As a HR staff, I want to update and modify an interview status so that I can scout talents that fit what the company wants.	UPDATE interviews SET Status = X WHERE InterviewID = X;

The `jobs` database requires the system to create a new entry whenever an employer posts a position. To guarantee that the job posting is correctly saved in the database, this should be completed as a transaction.

Here is our SQL Transaction:

BEGIN TRANSACTION;
    INSERT INTO jobs (EmployerID, JobTitle, JobDescription, JobCategory, Location, SalaryRange, PostingDate, ClosingDate, JobSlot, Status) VALUES (X, X, X, X, X, X, X, X, X, X);
    COMMIT;

3. Scenario 3: Job Management

Use Case/User Story	SQL Statements
As a candidate, I want to filter the job listings based on various criteria such as job title, job category, location, and salary range... so that I can search for a job that interests me.	SELECT * FROM jobs WHERE JobTitle LIKE X AND JobCategory = X AND Location = X AND SalaryRange >= X
As a candidate, I want to display only the most recent jobs so that I can take the current job market into consideration.	SELECT * FROM jobs ORDER BY PostingDate DESC LIMIT 50;

Since job searches simply involve reading data and don't alter the database's state, they usually don't require transactions.

4. Scenario 4: Application Management

Use Case/User Story	SQL Statements

As a candidate, I want to track the status of my applications so that I can be informed of my application status changes.	SELECT ApplicationID, Status FROM applications WHERE CandidateID = X;
As a HR staff, I want to view the applications received for my job postings so that I can read the profile of each candidate.	SELECT * FROM applications WHERE JobID IN (SELECT JobID FROM jobs WHERE EmployerID = X);
As a HR staff, I want to update and modify the applications received for my job postings so that candidates can plan interviews.	UPDATE applications SET Status = X WHERE ApplicationID = X;
As a HR staff, I want to reject applications when the job slot is full or the closing date has passed so that candidates don't waste their time signing for these positions.	UPDATE applications SET Status = 'Rejected' WHERE JobID = X AND ApplicationDate > X;

To maintain data consistency, updating an application's status should be done as a transaction. The `Status` in the `Applications` database must be updated if an application is denied.

Here is our SQL Transaction:

```
BEGIN TRANSACTION;
    UPDATE applications SET Status = X WHERE ApplicationID = X;
    COMMIT;
```

5. Scenario 5: Interview Management

Use Case/User Story	SQL Statements
As a candidate, I want to set notification for the interviews, so that I don't miss any important interview on my schedule.	SELECT * FROM interviews WHERE InterviewDate BETWEEN X AND X;
As a candidate, I would like an option to book an interview so that I can apply for a job.	INSERT INTO interviews (ApplicationID, CandidateID, JobID, InterviewDate, Status) VALUES (X, X, X, X, X);

A new record must be inserted into the `Interviews` table and the `Status` in the `Applications` table must be updated in order to schedule an interview. To guarantee data consistency, this needs to be completed as a transaction.

Here is our SQL Transaction:

```
BEGIN TRANSACTION;
    INSERT INTO interviews (ApplicationID, CandidateID, JobID, InterviewDate, Status) VALUES (X, X, X, X, X);
    UPDATE applications SET Status = 'Scheduled' WHERE ApplicationID = X;
    COMMIT;
```

6. Scenario 6: Training Course Management

Use Case/User Story	SQL Statements
As a candidate, I want to be able to search for courses based on various criteria such as course	SELECT * FROM trainingcourses

category, duration, and fee so that I can choose which training program is suitable for me.	WHERE CourseName LIKE X AND CourseCategory = X AND Instructor = X AND Duration <= X AND Fee <= X;
As a candidate, I want to track my progress in the enrolled courses so that I can expect to receive certificates upon completion.	SELECT * FROM enrollments WHERE CandidateID = X;

The system must add a new record to the `Enrollments` table whenever a candidate enrolls in a course. To guarantee that the enrollment is successfully saved in the database, this should be completed as a transaction.

Here is our SQL Transaction:

BEGIN TRANSACTION;
    INSERT INTO enrollments (CandidateID, CourseID, EnrollmentDate, CompletionDate, Status) VALUES (X, X, X, X, X);
    COMMIT;

7. Scenario 7: Authentication

Use Case/User Story	SQL Statements
As a candidate or a staff, I want to log in to the system by entering username and password so that the system can verify my credentials and authorize actions based on the user's role.	SELECT * FROM candidatecredentials WHERE Username = X AND Password = X; SELECT * FROM employercredentials WHERE Username = X AND Password = X;

Since user authentication merely entails viewing data, transactions are usually not necessary. In order to maintain data integrity, however, those actions might need to be carried out as a transaction if the system records login attempts or other user events.

Performance (indexes) (Week 9)

Based on our database's usage patterns, we made the strategic choice to create indexes on `CourseCategory` in the `TrainingCourses` table and `JobCategory` in the `Jobs` table.

By effectively reducing the amount of records or rows in a table that need to be inspected, indexes are used to speed up search queries. Even though each table in our dataset only has 1000 rows, adding an index on `JobCategory` and `CourseCategory` can greatly enhance the efficiency of these search queries if users often look for employment or training programs based on their categories.

It's crucial to remember that whereas indexes expedite search queries, write operations such as INSERT, UPDATE, and DELETE are slowed down. This is due to the fact that the index must be updated each time data is entered into the table. Overuse of indexes might result in higher storage requirements and slower writing speed.

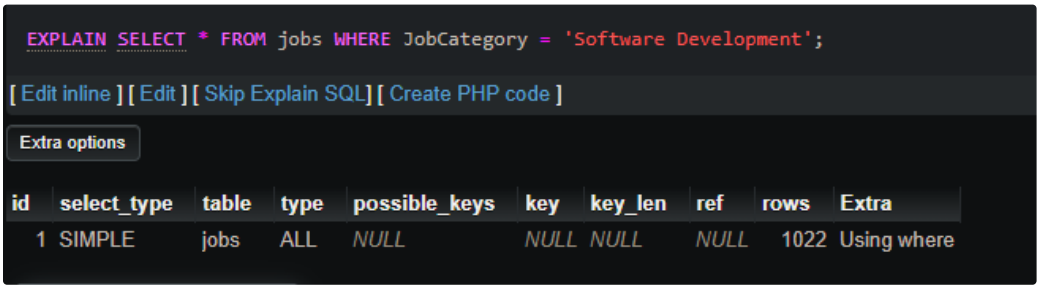
Therefore, since most of the users will want to look for job postings and training programs which are read operations (searches), it makes sense to optimize for that. Hence, creating indexes on `JobCategory` and `CourseCategory` would be beneficial.

Here are our SQL statements to create indexes on `JobCategory` in the `Jobs` table and `CourseCategory` in the `TrainingCourses` table:

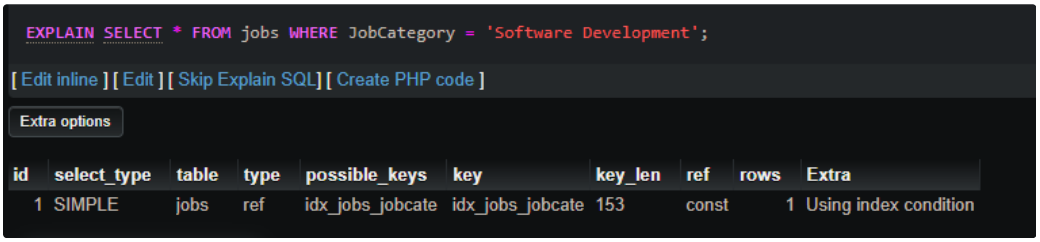
```
CREATE INDEX idx_jobs_jobcate ON jobs(JobCategory);
```

```
CREATE INDEX idx_trainingcourses_coursecate ON trainingcourses(CourseCategory);
```

We also monitored the performance of our database after creating indexes to ensure they are providing the desired results.



Without the index



With the index

```
EXPLAIN SELECT * FROM trainingcourses WHERE CourseCategory = 'Data Science';
```

[Edit inline] [Edit] [Skip Explain SQL] [Create PHP code]

Extra options

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	trainingcourses	ALL	*NULL*	*NULL*	*NULL*	*NULL*	1071	Using where

Without the index

```
EXPLAIN SELECT * FROM trainingcourses WHERE CourseCategory = 'Data Science';
```

[Edit inline] [Edit] [Skip Explain SQL] [Create PHP code]

Extra options

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	trainingcourses	ref	idx_trainingcourses_coursecate	idx_trainingcourses_coursecate	153	const	1	Using index condition

With the index

The constructed indexes "idx_jobs_jobcate" and "idx_trainingcourses_coursecate" were used by the database, as can be seen. In both scenarios, the table was not traversed more than required because only one row was scanned. The phrase "Using index condition" in the Extra column indicates that MySQL finished the filtering by utilizing the indexes.

The database could instantly return the single requested entry since it was able to skip scanning every row by using the index. In a small sample dataset such as ours, the index's use has little effect. However, retrieving the result required far less effort from the database, and this adjustment would have a big impact on a bigger dataset.

Team Reflection (Week 12)

📋 Overview

Reflect back on what you and your team learned and what motivates the group to succeed by following the instructions for the 4Ls Retrospective Play.

Team	Group 2.4
Team members	@Doan Hieu @ANH VU NGUYEN @HUY HOANG NGUYEN @MAI HANH PHAM @Nghiem Tuan Linh
Date	26/11/2023
Retrospective period	Progress Report (Week 4)

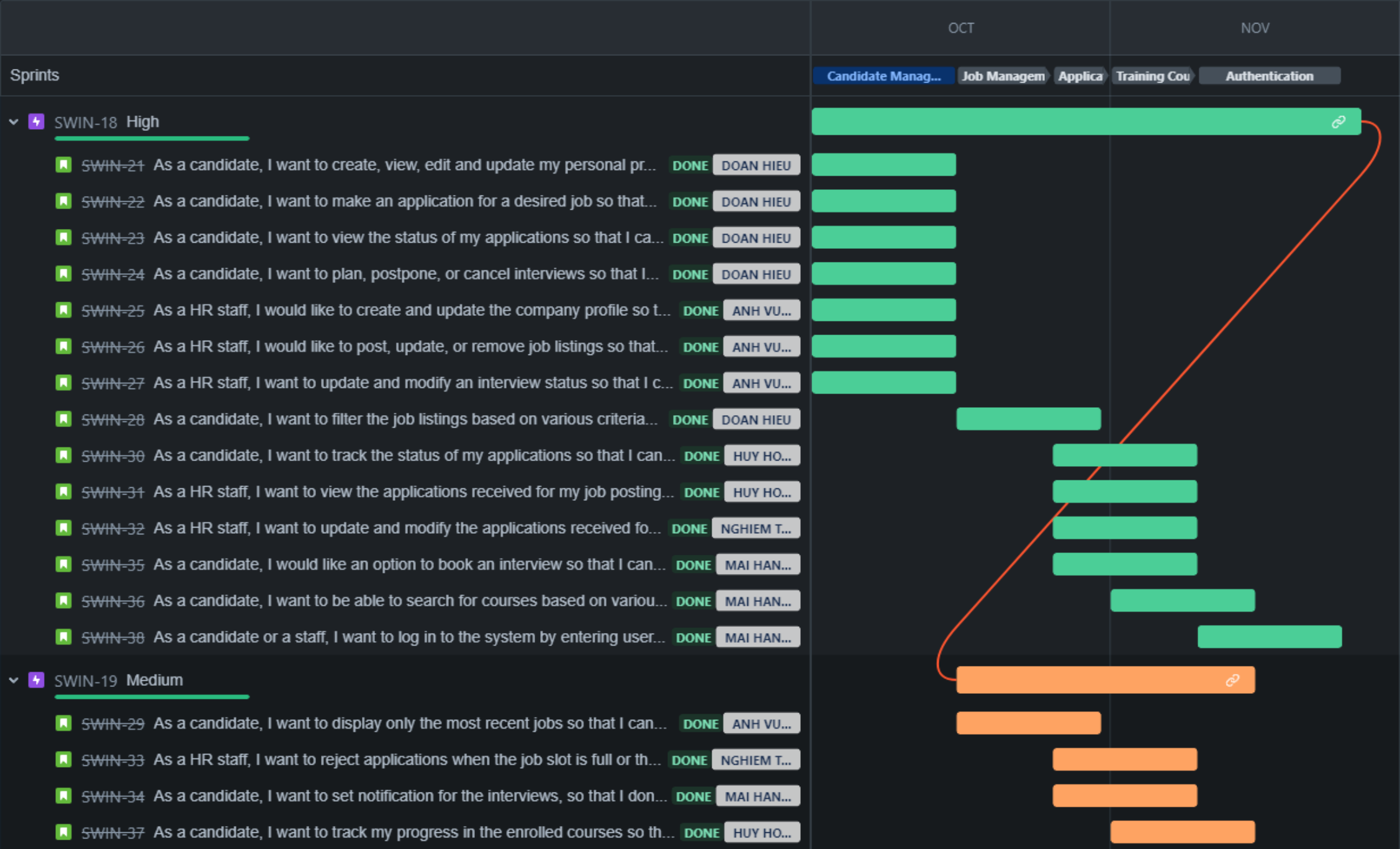
💭 4Ls retrospective

Milestones	Loved	Longed for	Loathed	Learned
Team Home page	Collaboration boosted efficiency	More interactive features for realtime updates	Formatting challenges while updating content	Communication is key for a unified team
Project Plan page	Detailed roadmap ensured clarity on project progression	Automated tracking tools for better project monitoring	Last minute adjustments disrupted the timeline	Flexibility is crucial to resolve challenges
Roles and Responsibilities page	Clearly defined roles streamlined workflow	Automated reminders for task deadlines	Occasional overlap in responsibilities caused confusion	Role clarity is essential but requires regular review
Risk Assessment page	Proactive identification of potential issues	Predictive analytics for more accurate risks	Some risks materialized despite mitigation efforts	Continual risk evaluation is crucial for project resilience
Persona page	Developing relatable user personas enhanced empathy	More diverse perspectives to personas	Time constraints limited in persona creation	Empathy driven design fosters user solutions
Empathy map	Visualizing user experiences deepened empathy	Real user interviews for more authentic insights	Some assumptions were not fit in the map	Iterative empathy mapping is vital for accuracy
Product requirements	Clear specifications development	Validation for instant feedback	Last minute changes caused rework	Detailed requirements development
Initial ER diagram	Visual representation clarified database structure	Tools for quick diagram adjustments	Some relationships were challenging to represent visually	ER diagrams evolve with understanding of the system

Appendix 1: Team member profiles	Insightful backgrounds create a strong team dynamic	Regular updates to profiles for evolving skill sets	Formatting inconsistencies across profiles	Understanding team strengths enhances collaborative success
Appendix 2: Team meeting note	Documentation help post meeting follow ups	Tracking for accountability	Missed detail and information due to short discussions	Efficient note taking is key for effective project execution

⚡Action plan

Action	Owner	Due date	Action items
Team Leader	@Doan Hieu	26/10/2023	Divide and assign tasks Internal meetings
Project Analyst	@ANH VU NGUYEN	30/10/2023	Calculating data Processing data evaluating progress
Secretary	@MAI HANH PHAM	30/10/2023	Managing correspondence Organizing meetings
Member	@Nghiem Tuan Linh	30/10/2023	Completing assigned tasks
Member	@HUY HOANG NGUYEN	30/10/2023	Completing assigned tasks


OCT
NOV
Sprints
Candidate Manag...
Job Managem
Applica
Training Cou
Authentication
SWIN-18 High
SWIN-21 As a candidate, I want to create, view, edit and update my personal pr... DONE DOAN HIEU
SWIN-22 As a candidate, I want to make an application for a desired job so that... DONE DOAN HIEU
SWIN-23 As a candidate, I want to view the status of my applications so that I ca... DONE DOAN HIEU
SWIN-24 As a candidate, I want to plan, postpone, or cancel interviews so that I... DONE DOAN HIEU
SWIN-25 As a HR staff, I would like to create and update the company profile so t... DONE ANH VU...
SWIN-26 As a HR staff, I would like to post, update, or remove job listings so that... DONE ANH VU...
SWIN-27 As a HR staff, I want to update and modify an interview status so that I c... DONE ANH VU...
SWIN-28 As a candidate, I want to filter the job listings based on various criteria... DONE DOAN HIEU
SWIN-30 As a candidate, I want to track the status of my applications so that I can... DONE HUY HO...
SWIN-31 As a HR staff, I want to view the applications received for my job posting... DONE HUY HO...
SWIN-32 As a HR staff, I want to update and modify the applications received fo... DONE NGHIEM T...
SWIN-35 As a candidate, I would like an option to book an interview so that I can... DONE MAI HAN...
SWIN-36 As a candidate, I want to be able to search for courses based on variou... DONE MAI HAN...
SWIN-38 As a candidate or a staff, I want to log in to the system by entering user... DONE MAI HAN...
SWIN-19 Medium
SWIN-29 As a candidate, I want to display only the most recent jobs so that I can... DONE ANH VU...
SWIN-33 As a HR staff, I want to reject applications when the job slot is full or th... DONE NGHIEM T...
SWIN-34 As a candidate, I want to set notification for the interviews, so that I don... DONE MAI HAN...
SWIN-37 As a candidate, I want to track my progress in the enrolled courses so th... DONE HUY HO...