CMPS-4131: SOFTWARE ENGINEERING - REPORT #2

	2
Group/Project	GR 2 - UB Job Portal

1	**Domain Analysis (25 pts)**		
a	Domain Model	8.00	8.00
	i. Concept Definitions	2.00	0.00
	ii. Association Definitions	2.00	2.00
	iii. Attribute Definitions	2.00	2.00
	iv. Traceability Matrix	2.00	2.00
b	System Operation Contracts	3.00	1.00
c	Data Model and Persistent Data Storage	3.00	2.00
d	Mathematical Model / Algorithms	3.00	0.00
2	**Interaction Diagrams (30 pts)**		
a	Diagrams	20.00	15.00
b	Description of Diagram	5.00	2.00
c	Alternate Solution Description #	5.00	0.00
3	**Class Diagram & Interface Specification (15 pts)**		
a	Class Diagram	5.00	5.00
b	Data Types and Operation Signatures	5.00	5.00
c	Traceability Matrix	5.00	5.00
4	**Algorithm and Data Structures(12 pts)**		
a	Algorithms	4.00	4.00
b	Data Structures	4.00	4.00
c	Concurrency	4.00	4.00
5	**User Interface Design and Implementation(4 pts)**		
a	Description (+)	2.00	2.00
b	Ease of Use (Usability)	2.00	2.00
6	**Test Case Design(15 pts)**		
a	List and Describe Test Case	5.00	4.00
b	Discuss Test Coverage / Integration Test	5.00	3.00
c	Plan for testing Algo, Non-Fnx Req. and UI Req.	5.00	3.00
7	**Project Management and Plan of Work(19 pts)**		
a	Merging the Contributions	10.00	10.00
b	Project Coordination	5.00	5.00
c	Plan of Work	2.00	2.00
d	Breakdown of Responsibilities	2.00	2.00
	PENALTY FOR LATE SUBMISSION		
	PENALTY FOR NO REFERENCE	**(-5)**	
	Total Value/ Points	120.00	94.00
	REPORT#2: VALUE/ GRADE	**10.00**	**7.83**
			78%

UB JOB PORTAL

Report #2

CMPS4131 - Software Engineering

Manual Medina

April 6, 2024

User Effort Estimation

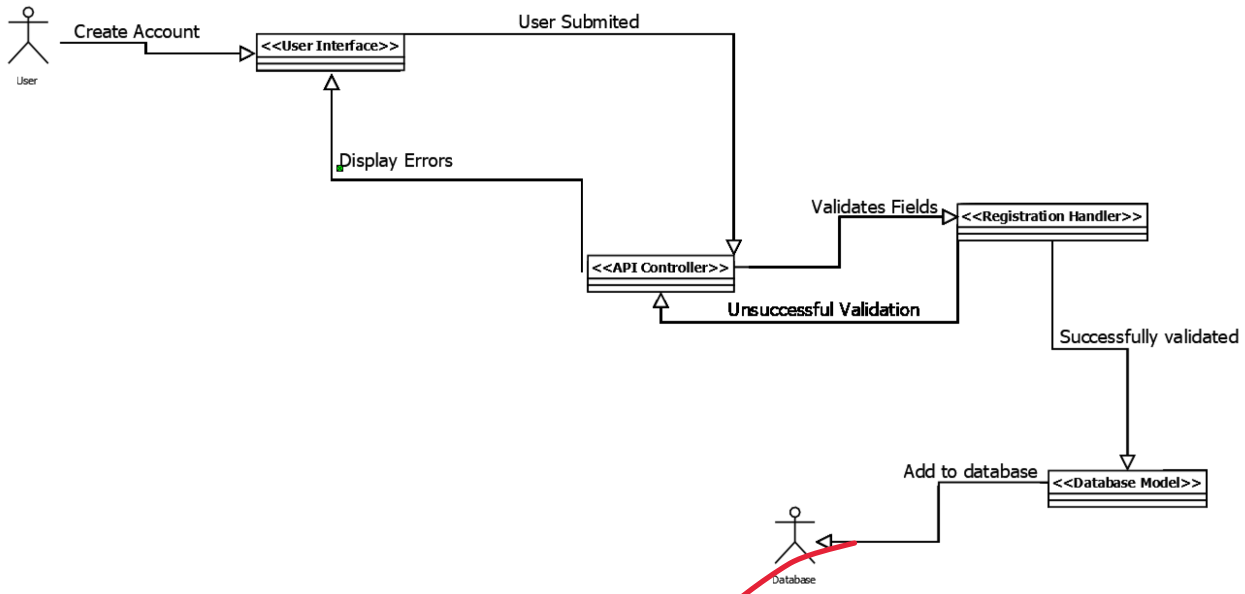
Responsibility Level		**Team Member Name**				
		Jahmur L	Jaheim L	Josias M	Tadeo B	Daniel M
	Project management				**100%**	
	Part 1 - Analysis and Domain Modeling					
	- Concept Model	75%			25%	
	- Concept Definitions		100%			
	- Association Definitions				100%	
	- Attribute Definitions			100%		
	- Traceability Matrix					100%
	- System Operation Contracts		100%			
	- Data Model and Persistent Data Storage	100%				
	Part 2 - Diagramming					
	- Interaction Diagrams			66.66%	33.33%	0%
	- Class Diagrams	0%		0%	100%	
	Part 3 - Design					
	- Algorithms and Data Structures					100%
	- User Interface and Implementation			100%		
	- Design of Tests	25%			75%	
	- Project Management and Plan of Work				100%	

Table of Contents

✓

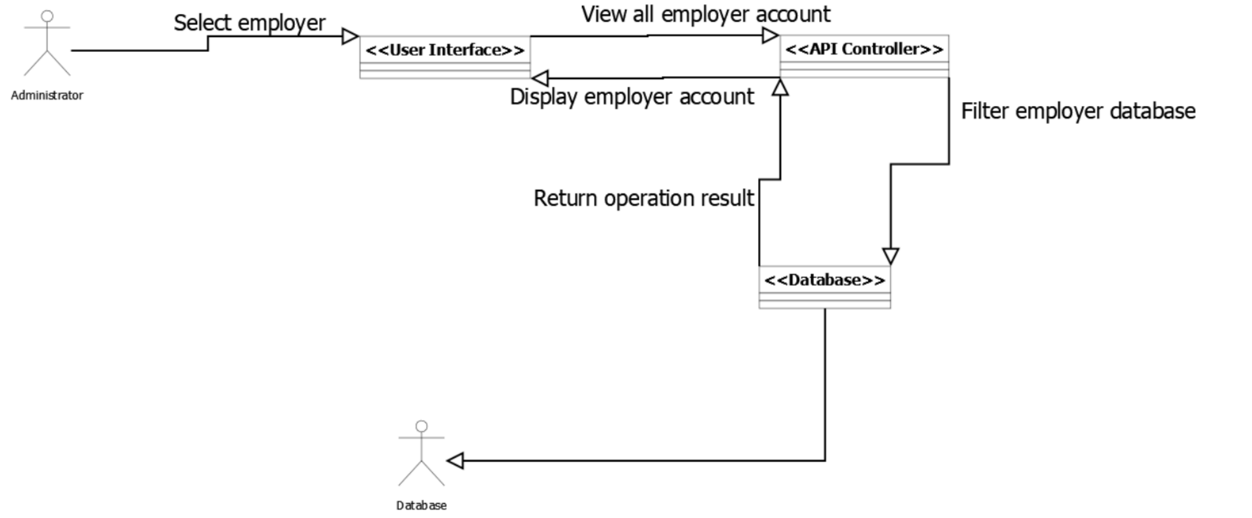
Analysis and Domain Modeling

Conceptual Model

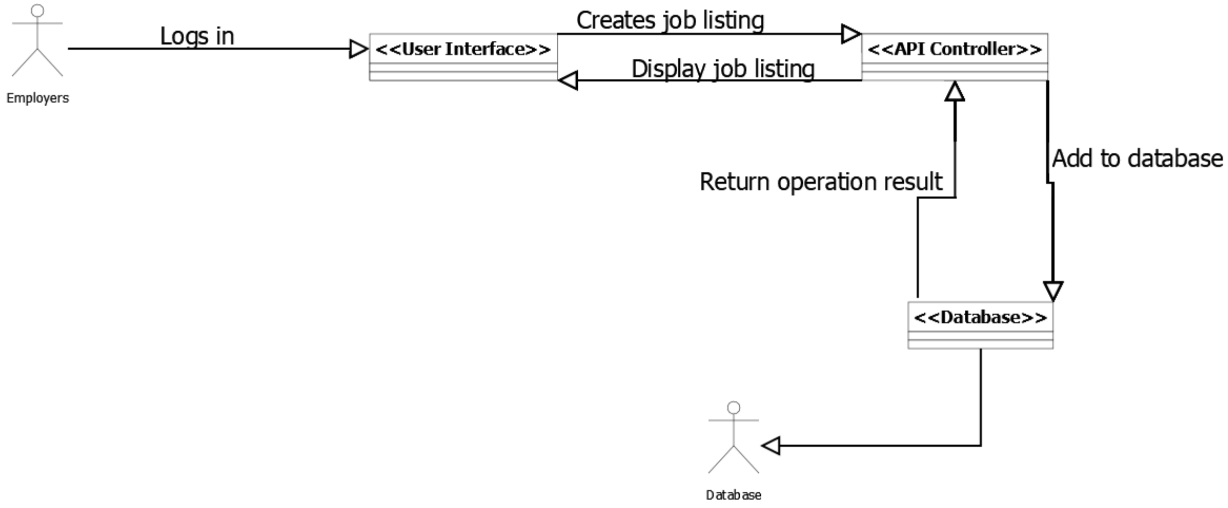


UC-1 Registration

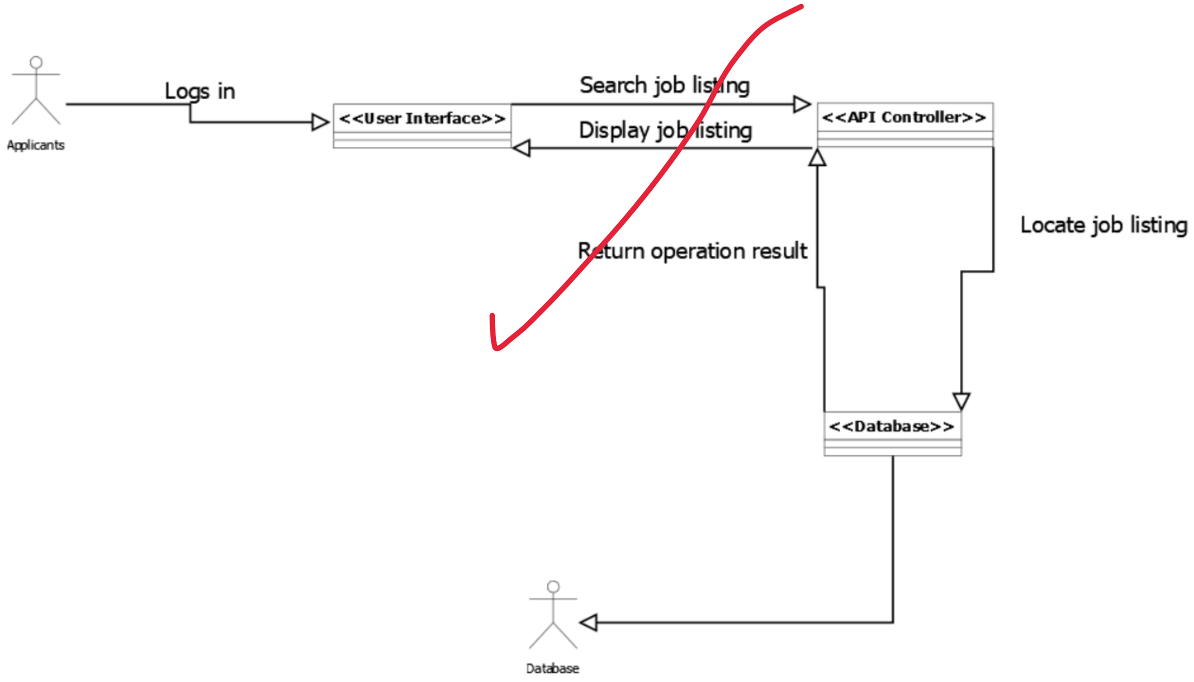
=



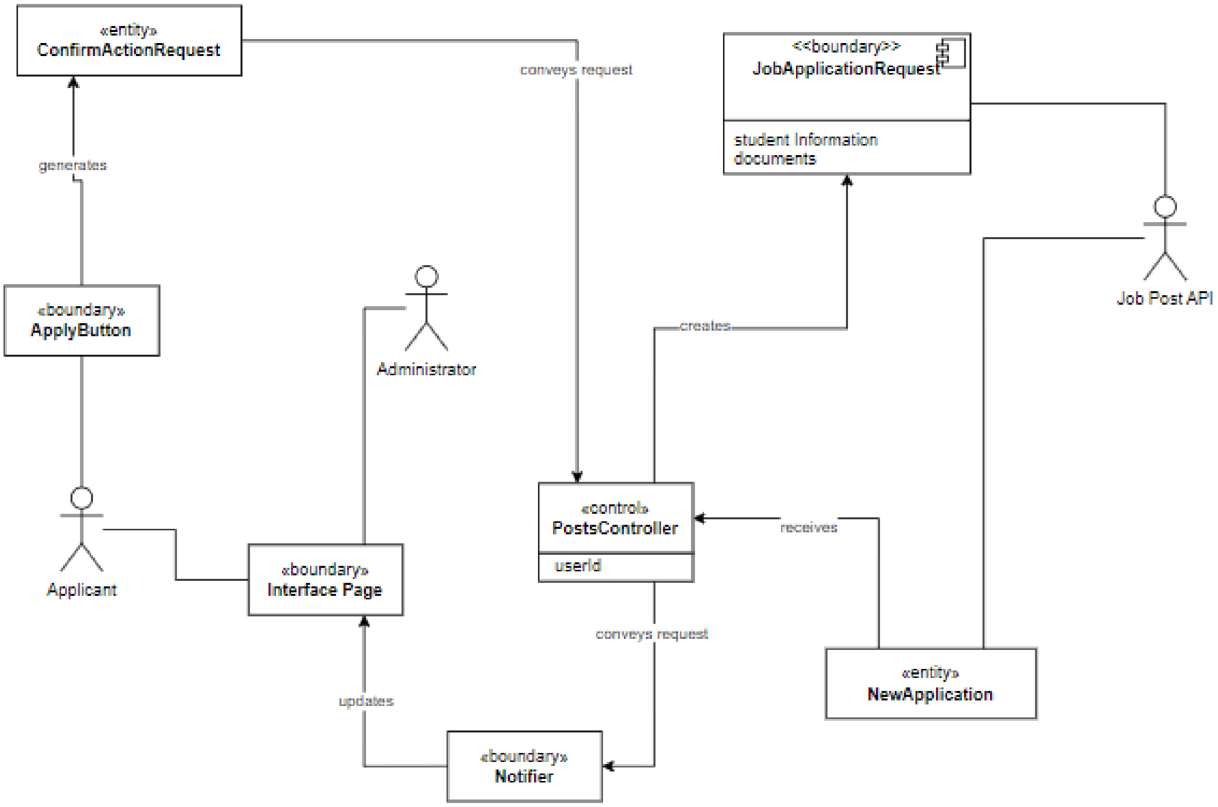
UC-2 Authentication



UC-5 PostJob



UC-6 SearchJobListing



UC-7 ApplyForJob

Concept Definitions

UC 1 - Registration

Responsibility Definition	**Type**	**Concept Definition**
Involves creating an intuitive and efficient interface that allows users to interact with the system to accomplish the task of registering use cases.		User Interface
Involves handling incoming requests from clients, processing the data, interacting with the backend services, and providing appropriate responses.		API Controller
Typically responsible for managing the actual process of registering new use cases within a system.		Registration Handler
Responsible for defining the structure and relationships of the data stored in the database related to use case registration.		Database Model

UC 2 - Authentication

Responsibility Definition	**Type**	**Concept Definition**
Responsible for capturing user credentials such as username, password, or any other form of authentication information.		User Interface
Acts as the gateway for handling authentication requests, implementing the necessary logic to verify user identities securely and efficiently.		API Controller
Serves as a critical component in the authentication process, storing and managing user authentication data securely while facilitating efficient and reliable authentication mechanisms.		Database

UC 5 - Post Jobs

Responsibility Definition	**Type**	**Concept Definition**
Typically involves providing an intuitive and efficient interface for users to input the necessary information and parameters related to posting a job.		User Interface
Revolves around handling incoming requests to create and post a job, processing the data provided, and orchestrating the necessary actions to fulfill the request.		API Controller
involves managing the storage and retrieval of job-related data, ensuring data integrity, and facilitating efficient querying and manipulation of job postings.		Database

UC 6 - Search Job Listings

Responsibility Definition	**Type**	**Concept Definition**
Involves designing an intuitive, efficient, and user-friendly interface that enables users to easily search and browse through job listings.		User Interface
Involves handling incoming requests from clients, processing those requests, interacting with the backend system or database to retrieve relevant job listings data, and providing an appropriate response back to the client.		API Controller
Involves storing and managing job listing data efficiently, facilitating fast and accurate retrieval of relevant listings based on search criteria provided by users.		Database

UC 7 - Apply for Job

Responsibility Definition	**Type**	**Concept Definition**
Trigger the action of submitting a job application.		Apply Button
Confirm the user's intent to proceed with a critical action, such as submitting a job application. This confirmation helps prevent accidental or unintended submissions.		Confirm Action Request
Should display relevant information about the job, including the job title, company name, location, job description, qualifications, responsibilities, and any other pertinent details. This presentation should be clear, organized, and easily understandable to the applicant.		Interface Page
Sends a confirmation email or notification to the applicant upon successful submission of their job application. This notification serves as a receipt, reassuring the applicant that their application has been received and is being processed.		Notifier
Responsible for receiving, processing, and storing job application data submitted by applicants. This involves validating the data to ensure it meets the required format and criteria, sanitizing inputs to prevent security vulnerabilities, and storing the data securely in the database.		Post Controller
Initiates the application process when a user expresses interest in applying for a job. This may involve capturing the user's action, such as clicking on an "Apply" button or selecting a job listing.		Job Application Request
Responsible for initiating the process when a user expresses interest in applying for a job.		New Application

Association Definitions

Use Case 1 - Registration

Concept Pair	Association Description	Association Name
User Interface ↔ API Controller	The user submits the account information for an applicant or employer and that information is passed to the API controller.	User Submitted
User Interface ↔ API Controller	The API controller returns any encountered errors to the user interface in the process of registration.	Display Errors
API Controller ↔ Registration Handler	The API controller forwards the information to the Registration Handler to validate the fields.	Validates Fields
API Controller ↔ Registration Handler	The Registration Handler returns the validation error to the API controller.	Unsuccessful Validation
Registration Handler ↔ Database Model	The Registration Handler interacts with the Database Model to create an account for the user.	Successfully Validated

Use Case 2 - Authentication

Concept Pair	Association Description	Association Name
User Interface ↔ API Controller	The admin submits the ID of the Employer they would like to delete.	Validate User
User Interface ↔ API Controller	The APIController returns an error if an invalid ID was provided.	Display error
API Controller ↔ Database Model	The APIController passes the ID in a request to the database to delete the employer with that ID.	Verify from database

API Controller ↔ Database Model	The Database returns the query result.	Return operation result

Use Case 3 - PostJob

Concept Pair	Association Description	Association Name
User Interface ↔ API Controller	The information for a new job post is sent from the user interface to the API controller.	Creates job listing
User Interface ↔ API Controller	The API controller returns the new job post for view in the employer feed and in the applicant’s job listings.	Display job listing
API Controller ↔ Database Model	The API controller requests that the information for a new job be added to the database.	Add to Database
API Controller ↔ Database Model	The Database Model is returns the status of the database operation.	Return operation result

Use Case 4 - SearchJobListing

Concept Pair	Association Description	Association Name
User Interface ↔ API Controller	An applicant enters the search criteria, and it is passed from the user interface to the API controller.	Search job listing
User Interface ↔ API Controller	The API controller returns the operation status and job records to the user interface.	Display job listing
API Controller ↔ Database Model	API Controller requests a job listing that matches the applicant’s criteria.	Locate Job Listing
API Controller ↔ Database Model	The Database job records that match the criteria is returned along with the operation status to the API Controller.	Return Operation result

Use Case 5 - ApplyForJob

Concept Pair	Association Description	Association Name
ApplyButton ↔ ConfirmActionRequest	The apply button generates a confirmActionRequest (with options to proceed or cancel)	generates
ConfirmActionRequest ↔ PostsController	The confirmActionRequest (when proceeding) saves the applicant identification information and forwards it to the PostsController.	Conveys request
PostController ↔ JobApplicationRequest	The PostController creates a JobApplicationRequest by getting the applicant's information using the identification information	creates
NewApplication ↔ PostsController	After a new application is created using the applicant's information, the PostsController receives this new application.	receives
PostsController ↔ Notifier	The PostsController returns the appropriate operation result for a successful application creation.	Conveys request
Notifier ↔ Interface Page	The notifier updates the administrator's interface page, notifying them of a new job application. Also, the applicant's interface is updated, notifying them that their application was sent.	updates

Attribute Definition

The UB Job Portal (UJP) is a software solution designed to address the challenges faced by both students and employers in Belize regarding the job application process. UJP facilitates efficient job searching, application tracking, and communication between applicants and employers.

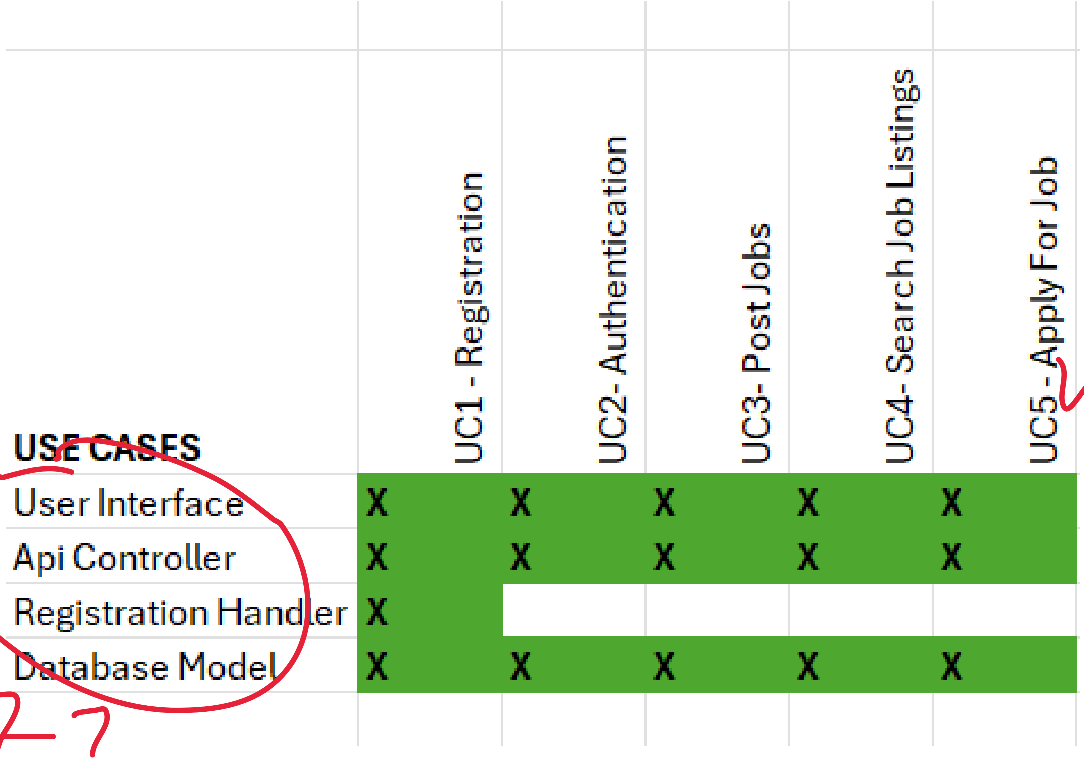
Key Features:

Efficient Job Search	UJP provides a centralized platform where students can explore job vacancies posted by various employers, eliminating the need to search through multiple sources such as social media, newspapers, and company websites.
Application Tracking (Student Perspective)	UJP enables students to manage and track their job applications seamlessly. Instead of manually updating Excel spreadsheets, students can rely on UJP to keep track of their application statuses and receive updates on their progress.
Application Management (Employer Perspective)	From the employer's standpoint, UJP streamlines the recruitment process by centralizing job vacancy posts and applicant information. Employers can easily review resumes, shortlist candidates, and communicate application updates to applicants through the platform.
Streamlined Communication	UJP facilitates efficient communication between employers and candidates. Employers can organize conversations with multiple shortlisted candidates and provide application updates with just a click, reducing the time spent on manual email correspondence.

Faster Document Handling	With UJP, handling documents and conversations becomes faster and easier for both employers and applicants. The system provides an organized way to view all sent and received documents, minimizing the hassle of managing attachments spread across multiple emails.
Enhanced Reach for UB Students	As a student or alumni of the University of Belize, users of UJP benefit from increased visibility to potential employers, improving their chances of securing employment opportunities.

Overall, UJP aims to improve efficiency and effectiveness in the job application process, ultimately contributing to the reduction of Belize's youth unemployment rate by connecting employers with UB candidates more quickly and facilitating seamless job search and application experiences for students.

Traceability Matrix:



USE CASES	UC1 - Registration	UC2- Authentication	UC3- Post Jobs	UC4- Search Job Listings	UC5- Apply For Job
User Interface	X	X	X	X	X
Api Controller	X	X	X	X	X
Registration Handler	X				
Database Model	X	X	X	X	X

7?

Table 1. Traceability Matrix Mapping use cases to domain concepts.

System Operation Contracts

Operation	StudentRegistration()
Cross References	UC 1 - Student Registration
Preconditions	● Students access the registration page through the web portal or app. ● The registration page is shown
Postconditions	● StudentRegistration() is called ● Function checks if the student is not already register for the course ● Once completed, the student registration details are updated into the database ● Notifier calls to print update on the

	screen

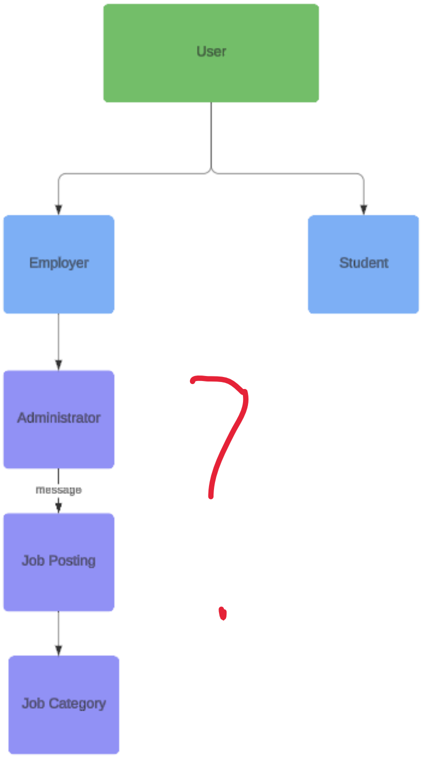
Operation	EmployerRegistration()
Cross References	UC 3 - Employer Registration
Preconditions	● Employers access the registration page through the web portal or app ● The registration page is shown
Postconditions Tense	● EmployerRegistration() is called ● Function checks if the employer is not already in the system ● Once completed, the employer registration details are updated into the database ● Notifier calls to print update on the screen

Operation	FilterJob()
Cross References	UC 5 - Filter Job
Preconditions	● Employer goes to the web page or app and logs in ● The employer goes to the filter job section
Postconditions	✓ ● FilterJob() is called ● Function should show a lost of all the job that have been filtered to meet the specific requirements ● If the filter gets change, then the job listings should be updated

Operation	UpdateCompanyInfo()
Cross References	UC 8 - Update Company Info
Preconditions	● Employer goes into the web page or app and logs in ● The page is presented ● Update company information is shown
Postconditions	● UpateCompanyInfo() is called ● Changes to the page will be made by Employer. ● All databases that's stored in the page will remain satisfied. ● Notifier will notify the changes made to the page.

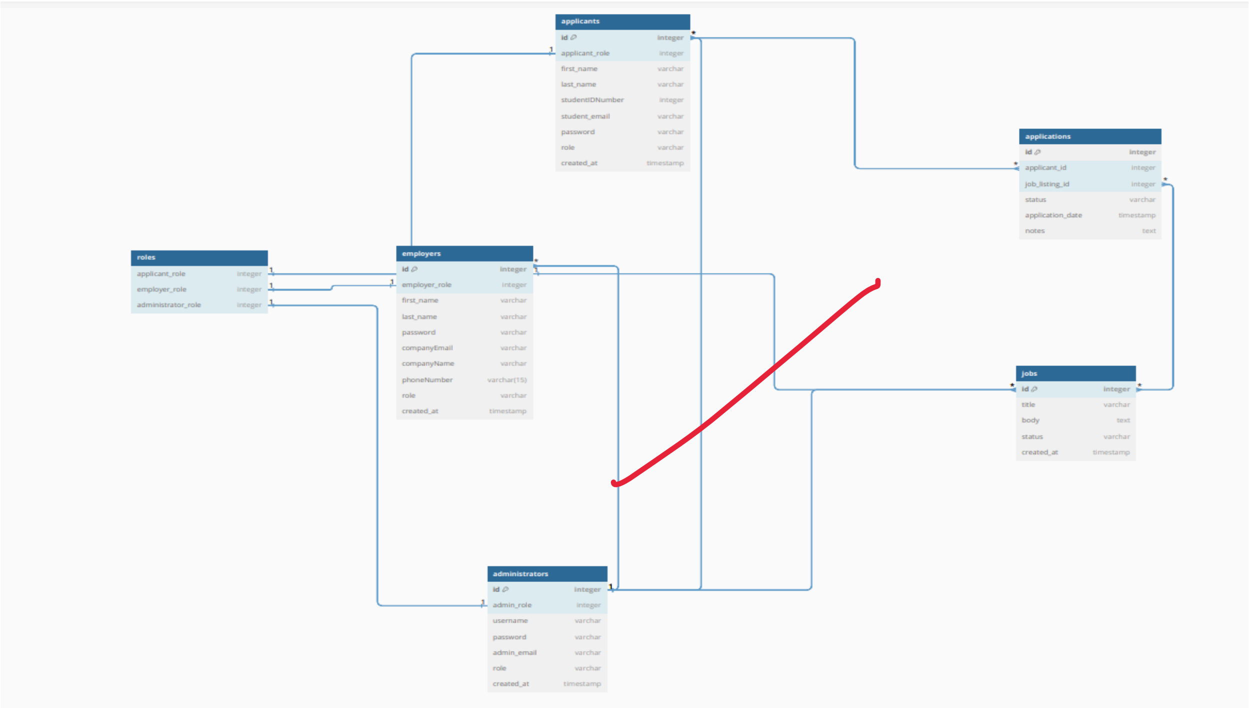
Operation	ChangeCompanyInfo()
Cross References	UC 9 - Change Company Info
Preconditions	● Employer goes into the web page or app and logs in ● The page is presented ● Change Company Information is shown
Postconditions	● ChangeCompanyInfo() is called ● Employer will make changes to the page. ● All databases that are stored on the

	page will remain satisfied. ● The notifier will notify the changes made to the page.



?

Data Models & Persistent Data Storage



applicants	
id	integer
applicant_role	integer
first_name	varchar
last_name	varchar
studentIDNumber	integer
student_email	varchar
password	varchar
role	varchar
created_at	timestamp

applications	
id	integer
applicant_id	integer
job_listing_id	integer
status	varchar
application_date	timestamp
notes	text

roles	
applicant_role	integer
employer_role	integer
administrator_role	integer

employers	
id	integer
employer_role	integer
first_name	varchar
last_name	varchar
password	varchar
companyEmail	varchar
companyName	varchar
phoneNumber	varchar(15)
role	varchar
created_at	timestamp

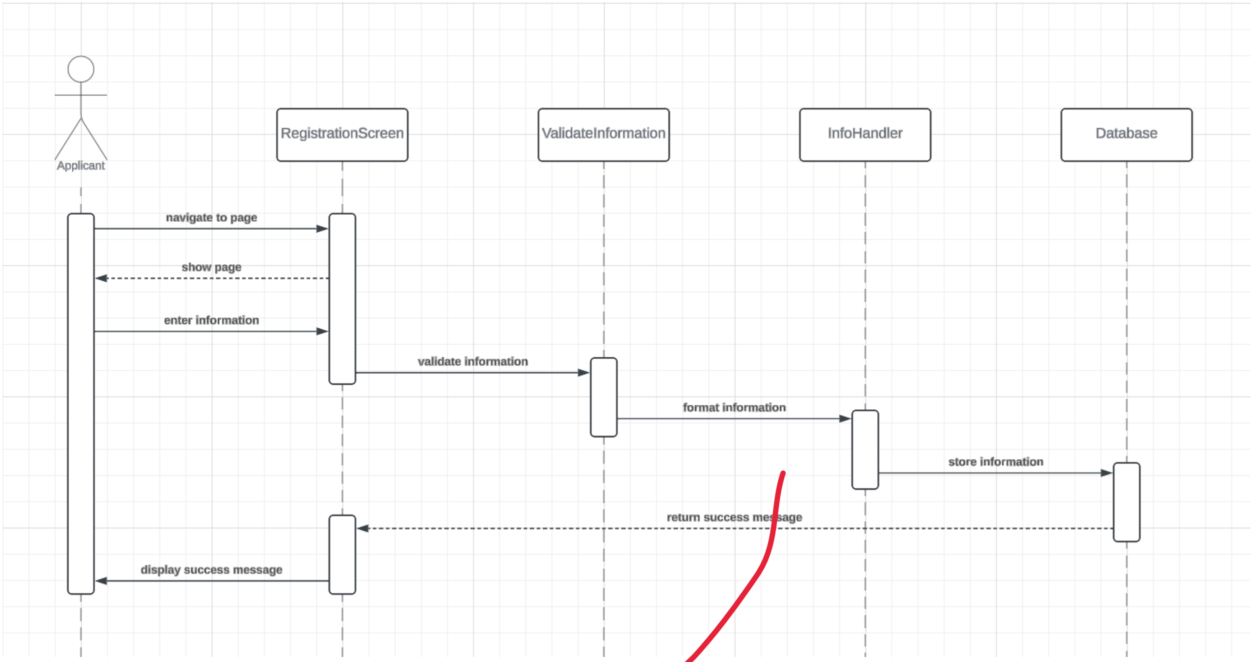
jobs	
id	integer
title	varchar
body	text
status	varchar
created_at	timestamp

administrators	
id	integer
admin_role	integer
username	varchar
password	varchar
admin_email	varchar
role	varchar
created_at	timestamp

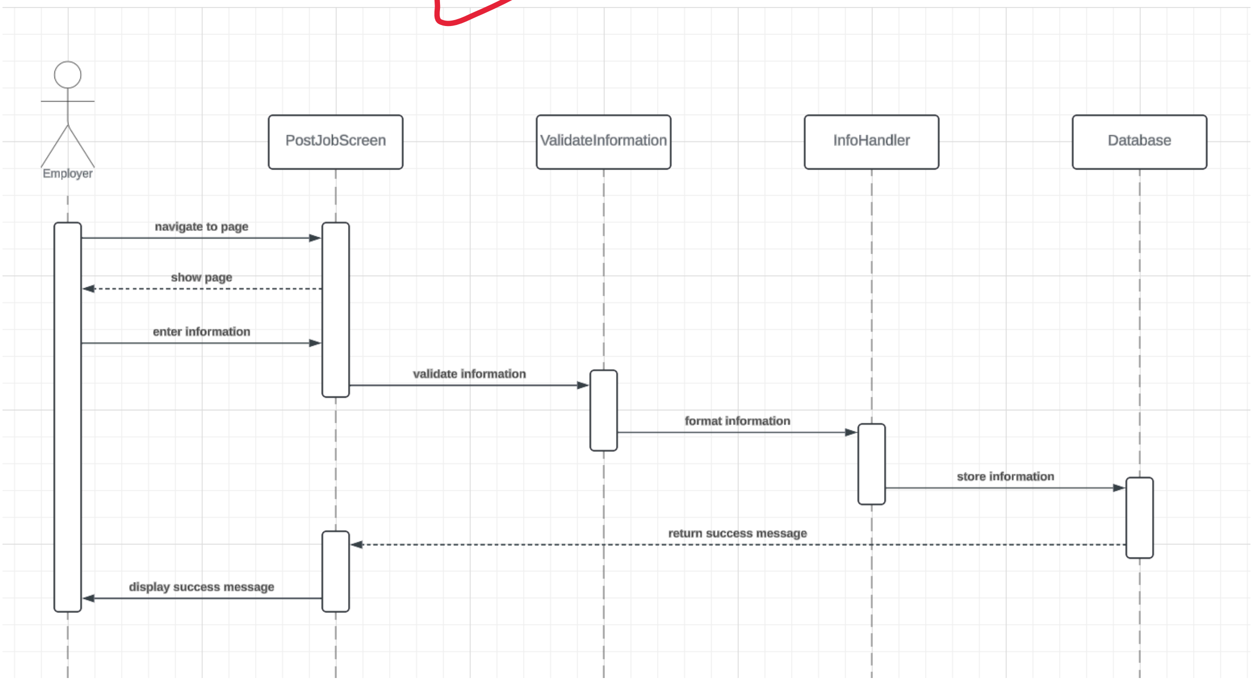
For this project, we do need persistent data storage. The data we need to have stored will be the login information from users and the job listings created by employers. We will do this by storing them in our database “Job Portal”. We will also need Admin and User data to have their own tables. We will be tracking which applications are being filled out by a student.

Interaction Diagrams

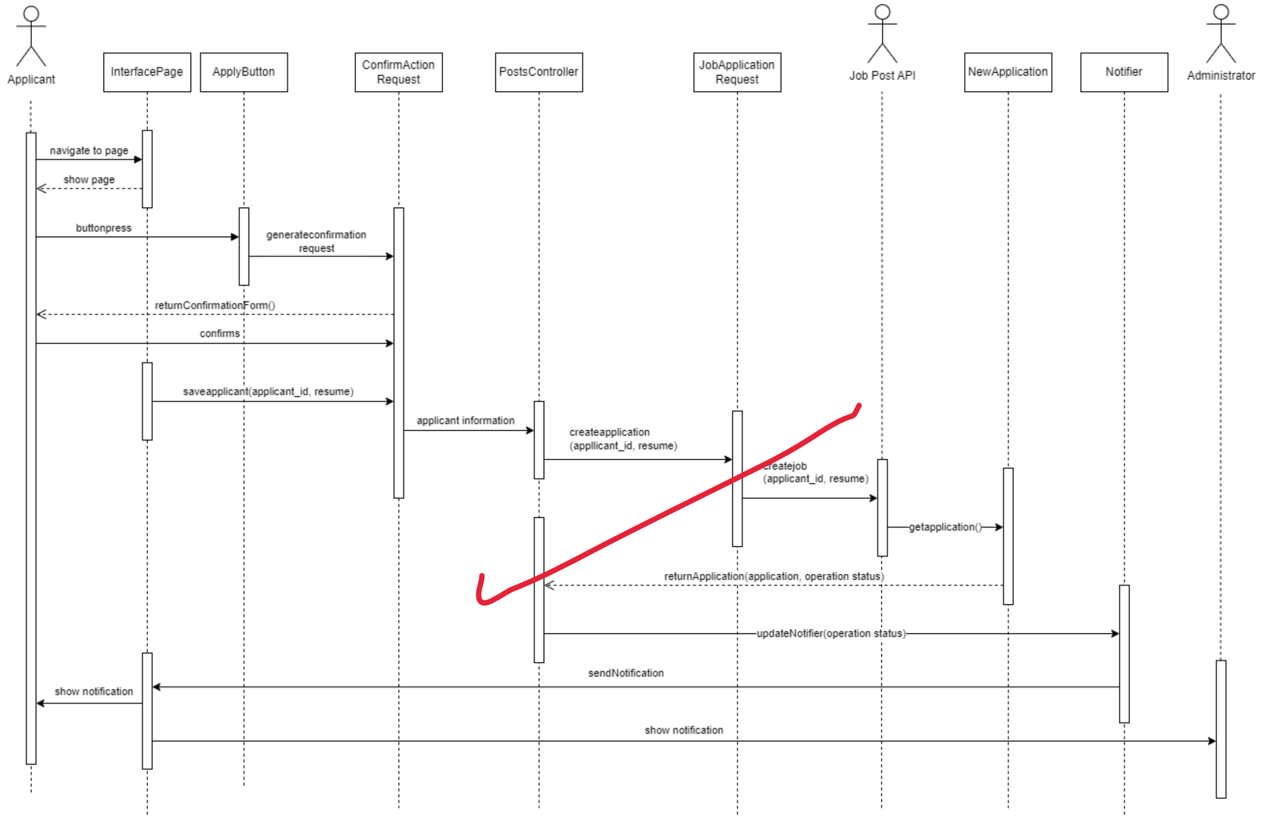
UC-1 Registration



UC-5 PostJob



UC-7 ApplyForJob

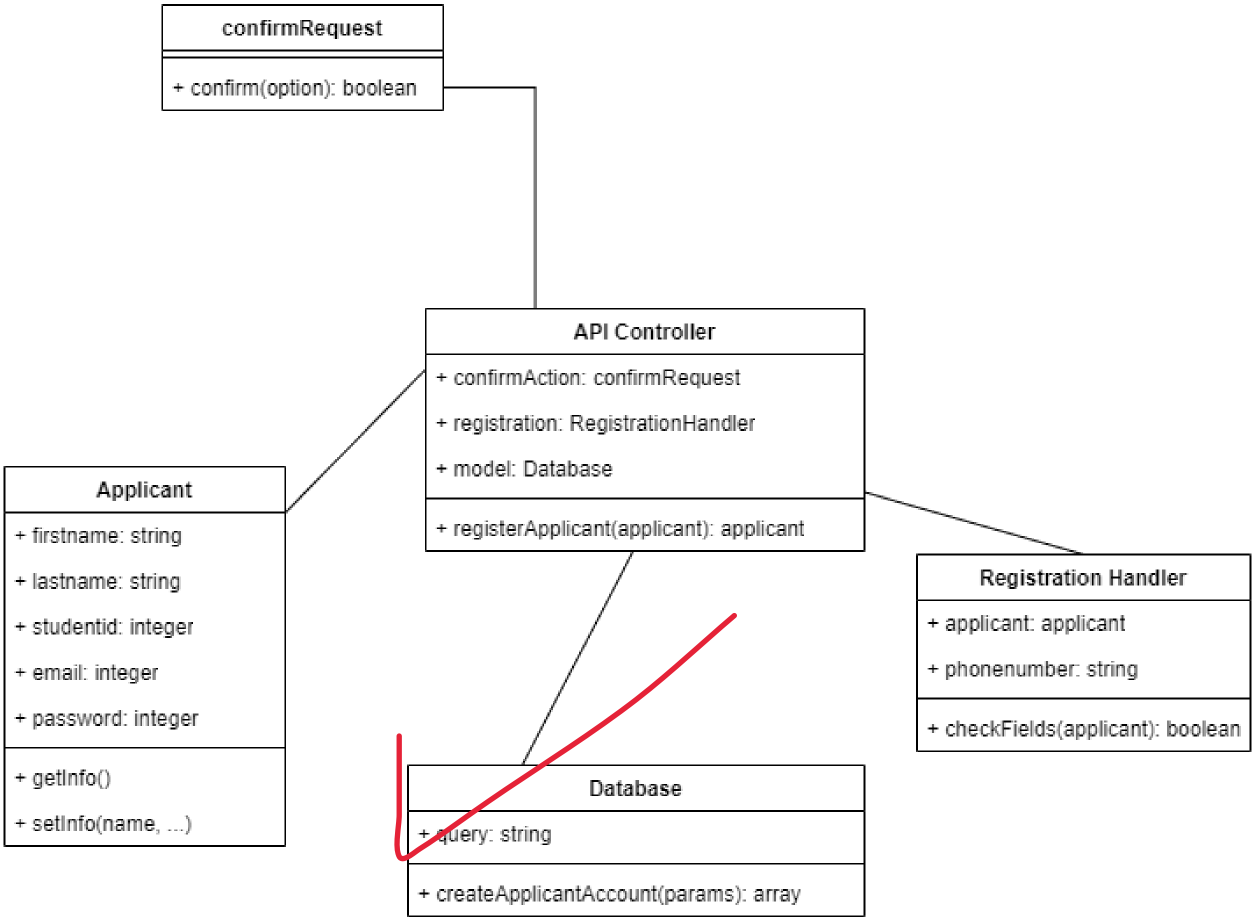


The applicant navigates to a URL, and the system returns the appropriate page. By clicking a button, the applicant generates a confirmationRequest which returns a form for a yes or no input. The user inputs a value of yes or no in a boolean representation, which is saved in the confirmationRequest. Upon confirmation, the information for the logged in applicant is forwarded to the PostsController. The PostsController passes the applicant information to the JobApplicationRequest which verifies the inputs. For database processing, the information is passed to an external API with the values of the user inputs for processing. The API returns the new information for a job application, and the system creates an application. The information for this application is forwarded to the PostsController. A notifier is updated with the system changes, and the notification is sent to the interface page for display to the applicant about a successful job application request. The administrator may also receive notifications about the new application for their posted job.

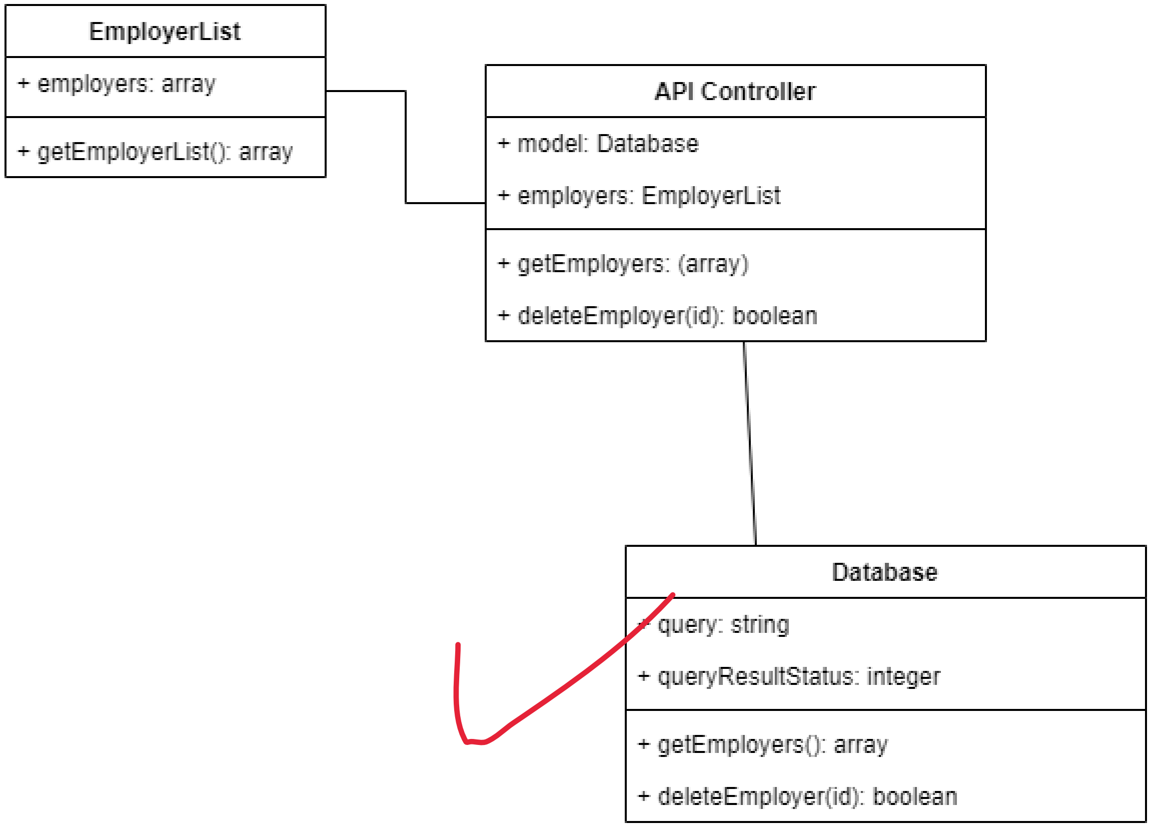
Class Diagram and Interface Specifications

Class Diagrams

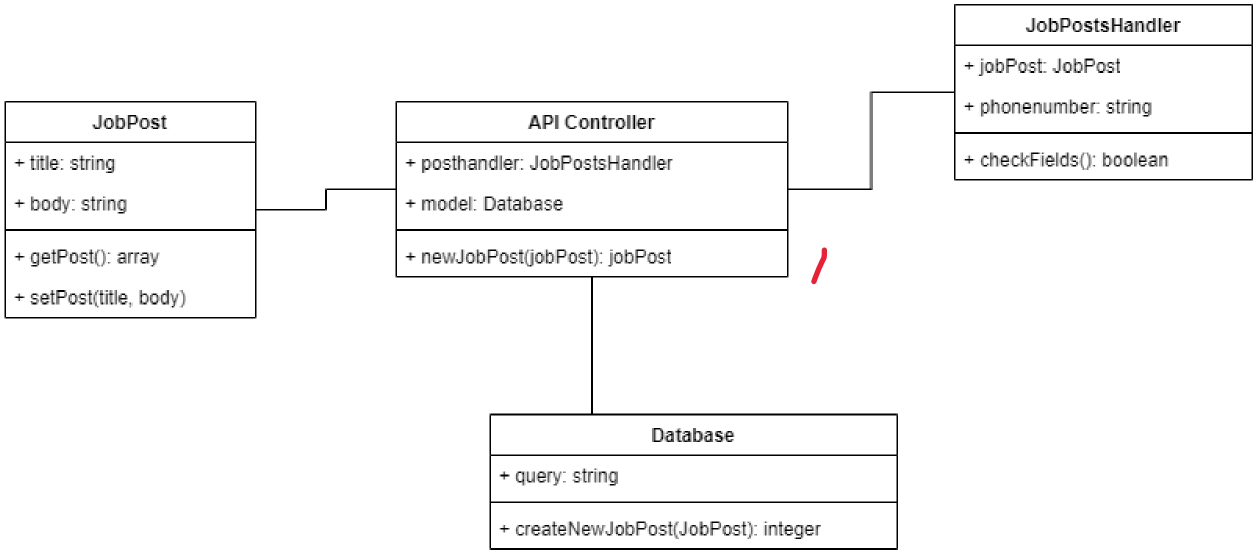
UC 1 - Registration



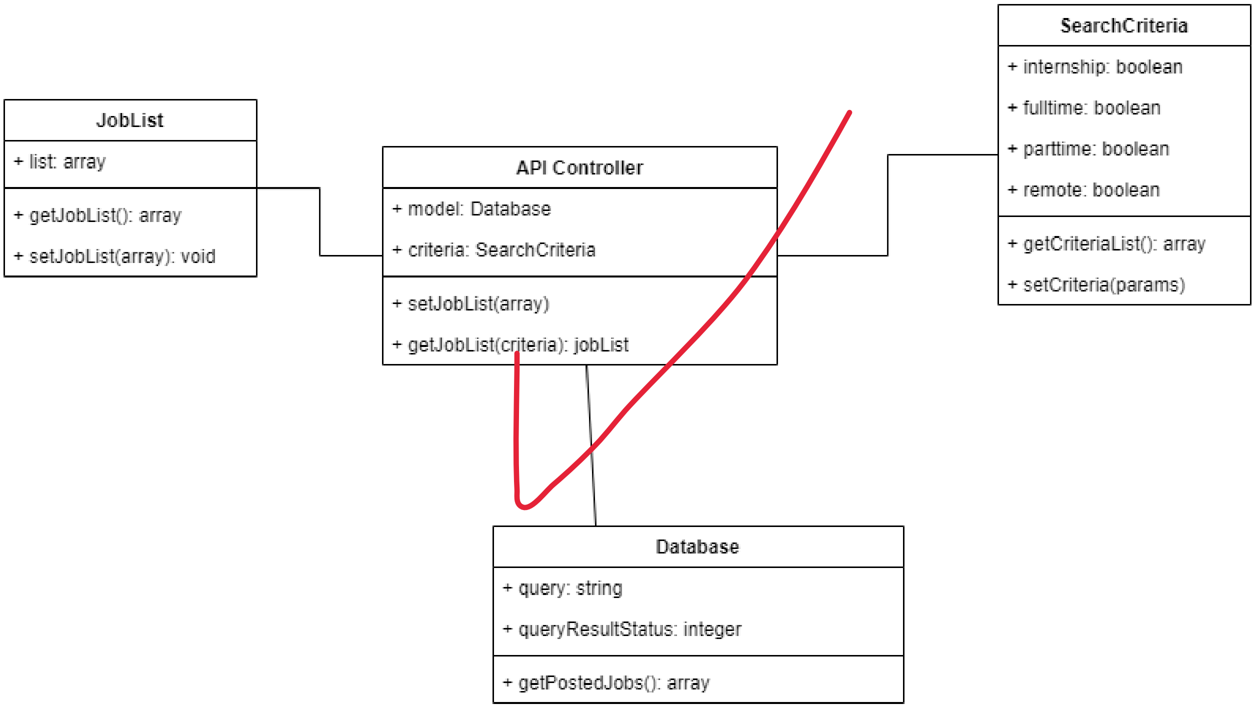
UC 2 - Authentication


EmployerList
+ employers: array
+ getEmployerList(): array
API Controller
+ model: Database
+ employers: EmployerList
+ getEmployers: (array)
+ deleteEmployer(id): boolean
Database
+ query: string
+ queryResultStatus: integer
+ getEmployers(): array
+ deleteEmployer(id): boolean

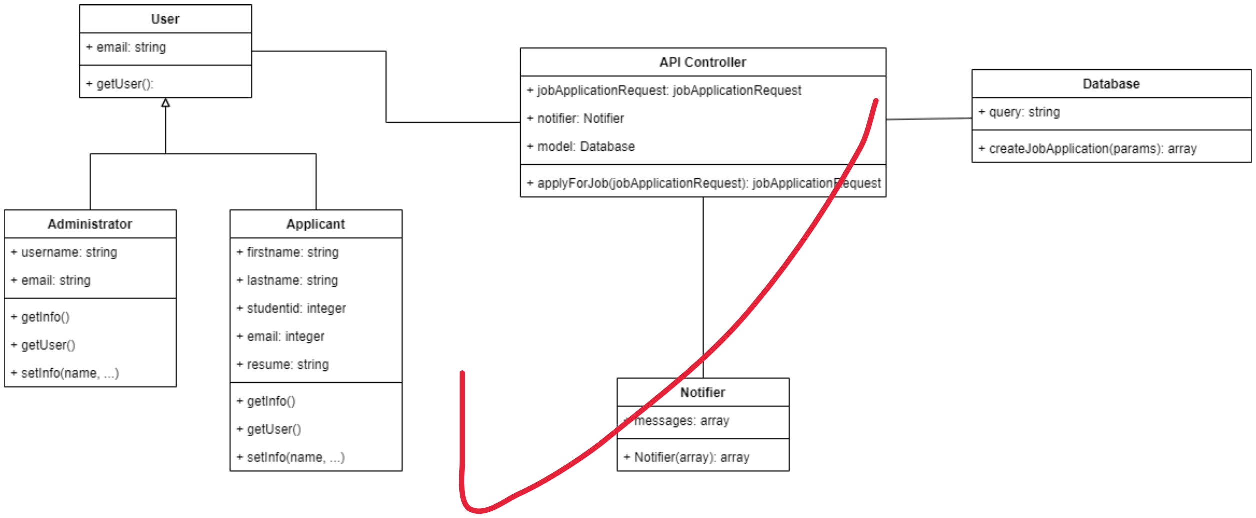
UC 5 - PostJob



UC 6 - SearchJobListing



UC 7 - ApplyForJob



Data Types and Operation Signatures

ConfirmRequest
Confirm(option): bool

Applicant
firstname: string lastname: string studentid: integer Email: integer Password: integer

Database
Query: string

Registration Handler
applicant: applicant phonenumber:string

EmployerList
employers:array

JobList
Title: string body: string

JobPostHandler
Jobpost: Jobpost phonenumber: string

SearchCriteria
Internship: bool Fulltime: bool Parttime: bool Remote: bool

User
email: string

Administrator
username: string email: string

Applicant
firstname: string lastname: string studentid: integer email: integer resume: string

Notifier
Messages: array

Traceability Matrix

Deriving classes from domain concepts.

APIController:
It is the primary controller for all the system's core functions. It is bundled with the PostsController to handle its functions. It is responsible for the flow of information between the user's interaction with the system and the database manipulation. It provides an easier and more structured way for developers to process data before the database handles it.

- UserInterface
- APIController
- PostsController
- Job Post API

Database
Handles the connection and manipulation of the database. It can also return values retrieved from the database, along with successful status messages or error messages, to other concepts.

- DatabaseModel (actor)

Applicant
Concerned with the primary actor(Applicant) who performs use cases in the system.

- Applicant

Administrator
Concerned with the primary actor(Administrator) who performs use cases in the system.

- Applicant

RegistrationHandler
Deals with necessary processes of registration for a system user. Interacts with other concepts to pass error messages. It also takes the load off the API controller.

- RegistrationHandler, APIController

ConfirmRequest
Deals with saving the user's option to confirm a particular action in the system's user interface

- UserInterface, APIController

EmployerList
Offloads the actions by handling the data structure that holds the list of employers retrieved from the database. It takes load off the APIController

- APIController

JobPost

Deals with saving the information for a particular job post.

- API Controller

JobPostHandler
Deals with necessary processes of validating a job post. Interacts with other concepts to pass error messages. It also takes the load off the API controller.

- APIController

SearchCriteria
Saves the information about the user’s selected search criteria from the user interface when filtering job posts. The SearchCriteria interacts with other concepts.

- API Controller, UserInterface, Database

Notifier
Saves the list of messages built during the process of carrying out functions to achieve a use case scenario. The messages are to be returned to the user interface for display.

- Notifier, UserInterface

Algorithms and Data Structures

The UB Job Portal system primarily implements algorithms for user authentication, job searching, filtering, communication, and data management:

User Authentication Algorithm:
Upon user registration, the system hashes and securely stores passwords using cryptographic algorithms like bcrypt or SHA-256.

During login, the system compares the hashed password provided by the user with the hashed password stored in the database to authenticate the user.

Job Search Algorithm:
When an applicant searches for jobs based on criteria such as job title, location, or industry, the system utilizes a search algorithm like binary search or linear search to retrieve relevant job postings from the database.
The search algorithm may incorporate indexing or caching techniques to improve search efficiency for large datasets.

Communication Algorithm:
For instant messaging between employers and applicants, the system employs a messaging algorithm that facilitates real-time communication, ensuring messages are delivered promptly and securely.

The messaging algorithm may utilize techniques like WebSocket protocol for bidirectional communication and message queuing for reliable message delivery.

Data Structures:
Yes, the system utilizes complex data structures such as arrays, linked lists, hash tables, and trees for efficient data management and retrieval:

Arrays: Used for storing and accessing fixed-size data elements such as user profiles, job postings, and application data.

Linked Lists: Employed for dynamic storage of user lists, job post histories, and communication logs, facilitating efficient insertion and deletion operations.

Hash Tables: Utilized for fast retrieval of user authentication information, job search results, and messaging metadata based on unique keys like user ID or Job IDs.

Trees: Employed for organizing hierarchical data structures such as category trees for job classifications, facilitating efficient search and traversal operations.

The choice of data structure depends on factors such as performance requirements like time complexity for search, insertion, deletion, memory usage, and flexibility in data manipulation.

Concurrency:
Yes, the system utilizes multiple threads to handle concurrent user interactions and background tasks such as data processing and messaging:

User Interface Threads: Separate threads of control are allocated for handling user interactions on the client-side, ensuring responsiveness and smooth user experience.

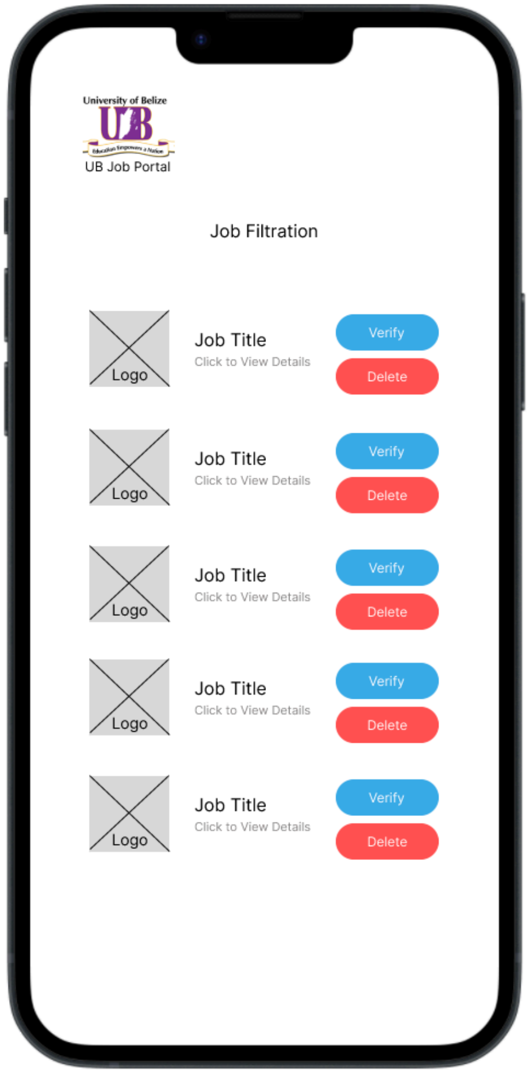
Server-Side Threads: Multiple threads are used on the server-side to handle incoming HTTP requests from clients concurrently, improving system scalability and throughput.

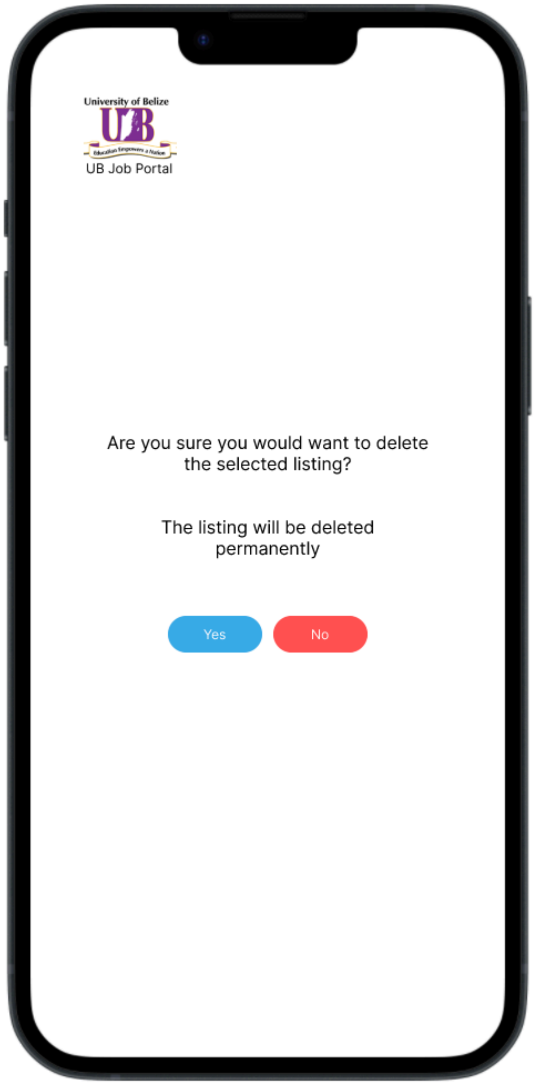
Synchronization: Synchronization mechanisms such as locks, semaphores, or mutexes are employed to ensure thread safety and prevent data races when accessing shared resources such as the database or message queues. Additionally, techniques like thread pooling may be used to manage and reuse threads efficiently.

User Interface Design and Implementation

FilterJob screens were updated to match the theme of other screens with the use of Figma, The structure of the screen was also updated to contain a list style allowing the user to filter through multiple job listings.

This should make it easier to manage multiple job postings at the same time overall making the process easier.





Are you sure you would want to delete the selected listing?

The listing will be deleted permanently

Design of Tests

List and describe the test cases that will be programmed and used for unit testing of your software

Test Case	TC-1
Use Case being used	UC1
Criteria for sucess/fail	The user registers their account, either as an applicant or employer.
Input Data	Alphanumeric
Test Procedure	Expected Result:
Step1: call function to create user ("", id, firstname, lastname, email, password, repeatedPassword) with user being of Employer or Applicant type.	Success
Step2: call function createController(user) by passing the created Employer or Applicant and saving it as a property/member.	Success
Step3: call function "registerUser(user)" in the controller by passing the saved user in the controller.	Success
Step4: call function "checkFields()" to check the validity of each input/value for the user type.	Success
Step5: call to function "passwordMatch()" with different passwords.	Fail
Step6: call function "checkForInvalidStringLengths()" on a created user object containing strings with invalid characters.	Fail
Step7: connect to the database	Success
Step8: call function "registerUserInDb()" to perform the requested action of inserting a user but with an invalid query string.	Fail
Step8: call function "registerUserInDb()" to	Success

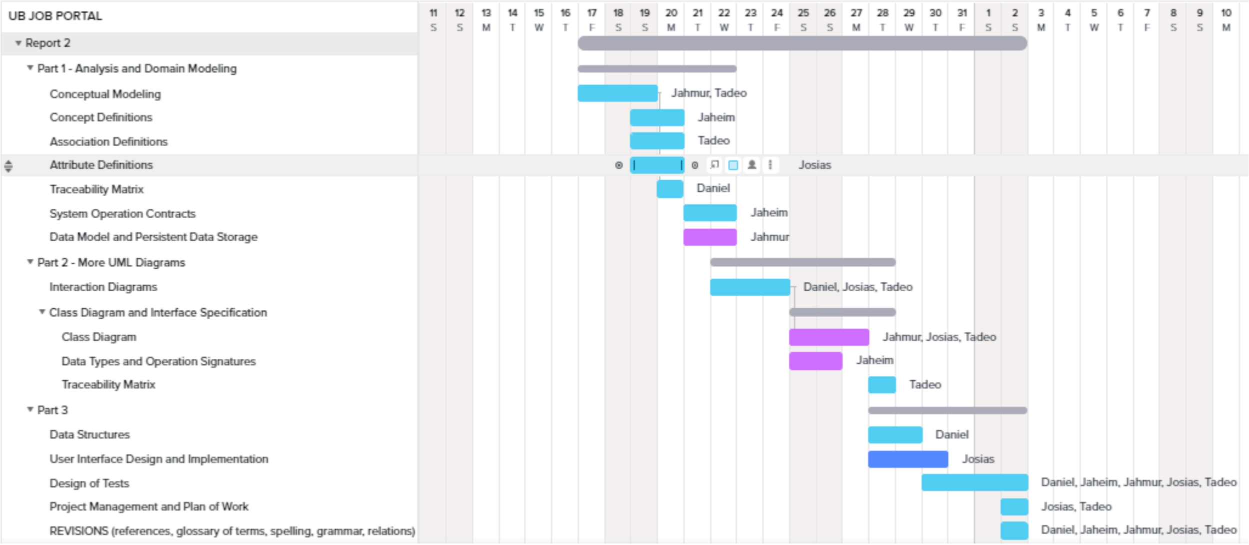
perform the requested action of inserting a user.	

Test Case	TC-2
Use Case being used	UC2
Criteria for success/fail	The administrator deletes an illegitimate Employer.
Input Data	Numeric
Test Procedure	Expected Result:
Step1: call function createController()	Success
Step2: call function “deleteEmploer(id)” in the controller by passing an numerical value as a string.	Success
Step3: call function “verifyId(id)” in the controller and pass a negative value	Fail
Step4: call function “verifyId(id)” in the controller and pass a value of 0	Fail
Step5: call function “verifyId(id)” in the controller and pass a valid ID	Success
Step6: connect to the database	Success
Step7: call function “deleteEmployerInDB()” to perform the requested action of deleting an employer using a valid ID.	Success

Test Case	TC-3
Use Case being used	UC5
Criteria for success/fail	The Applicant posts a job.
Input Data	Alphanumeric
Test Procedure	Expected Result:
Step1: call function to create jobPost(“”, jobtitle, jobdescription) on a jobPost object after creating it.	Success
Step2: call function createController(user)	Success
Step3: call function “newJobPost(jobPost)” in the controller by passing the created jobPost in the controller.	Success
Step4: create the jobHandler object and call function “checkFields()” to check the validity of each input/value for the jobPost type.	Success
Step5: call to function “checkJobPostContent()” with invalid job title.	Fail
Step6: call to function “checkJobPostContent()” with invalid job description.	Fail
Step7: connect to the database	Success
Step8: call function “createJobPostInDB()” to perform the requested action of inserting a job post but with an invalid query string.	Fail
Step9: call function “createJobPostInDB()” to perform the requested action of inserting a job post with a valid query string.	Success

Project Management and Plan of Work

Plan of Work



UB JOB PORTAL	
Report 2	
Part 1 - Analysis and Domain Modeling	
Conceptual Modeling	Jahmur, Tadeo
Concept Definitions	Jaheim
Association Definitions	Tadeo
Attribute Definitions	Josias
Traceability Matrix	Daniel
System Operation Contracts	Jaheim
Data Model and Persistent Data Storage	Jahmur
Part 2 - More UML Diagrams	
Interaction Diagrams	Daniel, Josias, Tadeo
Class Diagram and Interface Specification	
Class Diagram	Jahmur, Josias, Tadeo
Data Types and Operation Signatures	Jaheim
Traceability Matrix	Tadeo
Part 3	
Data Structures	Daniel
User Interface Design and Implementation	Josias
Design of Tests	Daniel, Jaheim, Jahmur, Josias, Tadeo
Project Management and Plan of Work	Josias, Tadeo
REVISIONS (references, glossary of terms, spelling, grammar, relations)	Daniel, Jaheim, Jahmur, Josias, Tadeo

References