

UB JOB PORTAL

Report #3



CMPS4131 - Software Engineering

Manual Medina

May 16, 2024

User Effort Estimation

R e s p o n s i b i l i t y L e v e l		Team Member Name				
		Jahmur L	Jaheim L	Josias M	Tadeo B	Daniel M
	Project management					
	Part 1 - Analysis and Domain Modeling					
	- Concept Model					
	- Concept Definitions					
	- Association Definitions					
	- Attribute Definitions					
	- Traceability Matrix					
	- System Operation Contracts					
	- Data Model and Persistent Data Storage					
	Part 2 - Diagramming					
	- Interaction Diagrams					
	- Class Diagrams					
	Part 3 - Design					
	- Algorithms and Data Structures					
	- User Interface and Implementation					
	- Design of Tests					
	- Project Management and Plan of Work					

Table of Contents

SUMMARY OF CHANGES	5
CUSTOMER STATEMENT OF REQUIREMENTS	6
Problem Statement	6-8
Glossary of Terms	9-10
SYSTEM REQUIREMENTS	11
Functional Requirements	11-12
Non-Functional Requirements	12-13
On-Screen Appearance Requirements:	13
FUNCTIONAL REQUIREMENTS SPECIFICATION	14
Stakeholders:	14
Actors and Goals	14
Use cases	15-16
Use Case Diagram	17
Traceability Matrix	18
Fully-Dressed Description	19-23
Sequence Diagrams	24-26
Interface Specification	27
Preliminary Design	27-30
User Effort Estimation	31-32
System Architecture	32
UML Package Diagram	32-33
Architecture Styles	34
Mapping Subsystems to Hardware:	35
Connectors and Network Protocols:	35
Global Control Flow	35
Hardware Requirements:	36
Plan of Work	37-38
Gantt Chart for the development of the System:	39-40
:Analysis and Domain Modeling	41
Conceptual Model	41-43
Concept Definitions	44-46
Association Definitions	47-49
Attribute Definition	50-51
Traceability Matrix:	51
System Operation Contracts	52-54
Data Models & Persistent Data Storage	55
Interaction Diagrams	56-57
Class Diagram and Interface Specifications	58
Class Diagrams	58-60

Data Types and Operation Signatures	61-62
Traceability Matrix	63
Algorithms and Data Structures	64-65
Design patterns	66
User Interface Design and Implementation	67
Design of Tests	68-70
Project Management and Plan of Work	71
Plan of Work	71
References	72

SUMMARY OF CHANGES

In Report 1 and 2, we initially presented class diagrams that were later found to contain several inconsistencies and omissions based on the feedback from our instructor. Specifically, there were errors in the relationships between classes, missing attributes, and unclear multiplicity notations. In response, we meticulously revised our class diagrams, ensuring all relationships are correctly defined, attributes are properly listed, and multiplicities are clearly indicated. The revised class diagrams now accurately reflect the system's structure and interactions.

Traceability Matrix

Our initial traceability matrix had gaps in linking requirements to their respective design and implementation artifacts. We expanded the matrix to include all requirements, ensuring each is traceable through use cases, class diagrams, and corresponding code modules. This enhancement ensures comprehensive coverage and better traceability throughout the project's lifecycle.

Use Cases

We refined our use cases to address ambiguities and gaps highlighted by our instructor. This involved clarifying the primary and alternate flows, specifying preconditions and postconditions more precisely, and ensuring that each use case is consistently aligned with the corresponding functional requirements. These improvements have resulted in more robust and detailed use cases that better guide the development process.

Diagrams

In addition to the class diagrams, we also made significant updates to our sequence and activity diagrams. These updates addressed issues such as missing interactions, unclear transitions, and incomplete sequences. The revised diagrams now provide a more accurate and detailed visualization of the system's dynamic behavior, facilitating better understanding and implementation of the system's functionality.

Project Code Adjustments

Based on the revised diagrams and use cases, we made several adjustments to our project code. These changes were essential to align the implementation with the updated design specifications.

This included refactoring classes, updating method signatures, and revising the logic to match the corrected use cases and diagrams. These code modifications ensure that the implementation accurately reflects the intended design and functionality.

Additional Improvements

Error Handling: Enhanced error handling mechanisms throughout the system, ensuring more robust and user-friendly error messages.

Documentation: Updated all project documentation to reflect the changes made in class diagrams, use cases, and code. This includes inline comments, user manuals, and developer guides.

Testing: Expanded our test cases to cover new and updated functionalities, ensuring comprehensive testing and validation of the system.

By incorporating the feedback from our instructor and making the necessary corrections, we have significantly improved the quality and accuracy of our project. These changes not only address the issues identified in previous reports but also enhance the overall robustness and maintainability of our system.

CUSTOMER STATEMENT OF REQUIREMENTS

Problem Statement

Finding a job is hard. The problems students face

In Belize, it is common knowledge that finding a job after university, given your experience, is quite hard. To find a job, students must scour social media, newspapers, and company websites to find a vacancy in an organization. In addition, some students rely on asking friends, family, and others to learn about available jobs. This old way of finding a job is time-consuming and unproductive. Given the right tool, the time it takes to find a job vacancy can be used to apply for appropriate jobs. Other popular sites like Indeed or LinkedIn are great for individuals to post their resumes and more information about themselves. Also, these applications have integrated features to allow individuals to learn more about posted job vacancies easily. Unfortunately, many organizations in Belize still rely on older methods to get out their job applications and hire new candidates. Henceforth, a more modern but “Belizean” solution must be implemented to encourage organizations to use these new technologies further to enhance their search for the best student candidates. Furthermore, it helps students find jobs that best suit them.

Keeping track of applications (student perspective)

There is no guarantee of success in one job application in the current job market. Therefore, it is in their best interest that a student applies for as many jobs as possible. Remembering and revisiting the numerous applications and tracking their status can be unmanageable. Such tracking can be done using Excel spreadsheets. The drawback comes with the limiting factor of manually updating new rows of information, which could get numerous and frustrating. An integrated tool can make this feature seamless and more intuitive.

Keeping track of applications (employer perspective)

Places companies post vacancies include their websites, newspapers, or social media such as Facebook. In today's market, youth unemployment is 21.81% of the workforce aged 15 - 24 in 2022 (*Belize Youth Unemployment Rate 1991-2024*, n.d.). Therefore, we can deduce that a job application on social media can have numerous applicants. Usually, these applicants would have to submit their resumes through email, as this is the norm for professional communication in Belize. The employer must then examine the resumes and CVs provided through email. A short list is created with the individuals to interview. The employer is then obliged to respond to all the other applicants about the status of their resumes. In some cases, applicants are not told the status of their application and take the initiative to reach out, requesting an update. This whole process can become efficient for both applicants and employers by using a centralized job vacancy application. In such an application, employers can manage all job vacancy posts with related applicant information shortlists and issue application updates to candidates not considered. This speeds up the process of employers finding a suitable candidate and does not leave other candidates guessing if their resume was considered.

Companies may enjoy the organization of speaking with multiple candidates for a posted job

Another advantage of a centralized system is that a company can organize conversations between numerous shortlisted candidates about a particular vacancy. This streamlines the process of communication between employers and potential candidates without the stress of email conversations. As mentioned above, candidates not considered for the job can be easily updated through an employer's click of a button, removing the time to update each candidate individually.

Companies can reach more people and different kinds of people much faster

Companies sometimes need to find a suitable candidate quickly to fill an important role. Not all companies have the reach or the following to discover the best candidates. With the advent of a

new job portal application for UB students, companies can leverage this tool to get their job vacancies directly in the eyes and interests of hundreds of students. Job posts can instantly reach candidates with different fields of expertise, degrees, and experience. This, in turn, minimizes the time an employer spends searching for a candidate. This also increases the chances of finding the perfect candidate for the role. The system puts the employer and candidate in direct contact with each other. An employer can converse with potential candidates, only taking the conversation to other applications if a video call is required.

Sending and receiving documents become faster and easier

Handling numerous documents and conversations through Gmail (the most common communication method in this process) can become messy, especially when handling numerous attachments spread across multiple messages in an email thread. Henceforth, an organized way to view all sent and received documents between parties can make processes for employers more intuitive. Such a feature is incredibly useful and helpful for employers, especially in limited time.

Advantages for UB students

As a student and alumni of the University of Belize, these students automatically have the advantage of reach when it comes to getting their skills, experiences, and resumes out to potential employers. This may improve the likelihood of UB students getting hired.

In conclusion, an application built to address these problems can ultimately impact Belize's youth unemployment rate. It allows employers to reach UB candidates faster and allows students to find and apply for multiple jobs easily.

Glossary of Terms

Term	Definition
Employer	The entity that makes job vacancy posts and considers candidates for hire.
Applicant	A UB student who applies for posts seeking an interview for hire.
Application	Submission of one's resume or other personal details to be considered for an interview or job.
Email Thread	A group of emails in an email conversation between two Gmail users.
Administrator	The user that had admin rights to verify job listings, verifying legitimacy of vacancies and users registered.
Analytics	The website will be able to
Client-Server Model	A computing architecture where tasks or services are divided between service providers (servers) and service requesters (clients). Clients initiate requests for services, while servers respond to those requests by providing the requested resources or performing the requested tasks, enabling distributed computing over a network.
Relational Database	A relational database is a type of database that organizes data into tables of rows and columns, with relationships defined between the tables, enabling efficient storage, retrieval, and manipulation of structured data.
REST API	REST API is an architectural style for an API that uses HTTP requests to access and use data, and exchange it securely over the internet. REST API is a way for two computer systems to communicate.
RESTful API	RESTful API is an interface that allows two different systems to exchange information over the internet with tight security. RESTful APIs is the web-service implementation of the REST architectural style and offers a scalable and simple method to construct APIs that are

	applicable to various programming languages and platforms.
CRUD	The CRUD (Create, Read, Update, Delete) paradigm is popular in web application development because it provides memorable foundation for reminding developers how to create com

SYSTEM REQUIREMENTS

Functional Requirements

Priority Weight	Description
1	Not important
2	Low importance
3	Normal
4	Important
5	Very Important

Identifier	PW	Requirement
REQ-1	2	The system shall allow applicants and employers to register an account to access the UB Job Portal.
REQ-2	3	The system shall provide applicants with CRUD (Create, Read, Update, Delete) functionality to manage their accounts, including the ability to upload personal information and resume documents, as well as edit their profiles.
REQ-3	5	The system shall allow employers to create vacant job posts.
REQ-4	3	The system shall allow employers to perform CRUD (Create, Read, Update, Delete) operations to manage their accounts, including the ability to upload company information and edit their profiles, including company information and job posts.
REQ-5	5	The system shall allow applicants to search for job posts based on criteria.
REQ-6	2	The system shall allow applicants to filter job post results based on preferences such as full-time/part-time, internship, or remote positions.
REQ-7	2	The system shall allow applicants to apply for job posts.
REQ-8	1	The system shall provide employers access to view applicant information and documents.
REQ-9	2	The system shall facilitate communication between employer and applicants, allowing for instant messaging in regard to a job post.

REQ-10	3	The system shall notify applicants about new and related job posts and their application status as well as notify employers about related prospective applicants.
REQ-11	3	The system shall provide reporting and analytics capabilities to track usage metrics, such as the number of job postings, applications, and user interactions.

Non-Functional Requirements

Functionality

NONREQ-1: The system should use a method to move user data to the database and back to the user.

NONREQ-2: The system will be responsible for saving data from electronic forms into a database.

NONREQ-3: The platform shall provide companies with the ability to showcase and upload job vacancies.

NONREQ-4: Students can browse, search, and view available job opportunities.

NONREQ-5: Companies and applicants should be able to register, creating individual profiles.

Usability

NONREQ-6: The system should be a clear, user-friendly, and understandable web app.

NONREQ-7: The platform should have an intuitive and user-friendly interface for both companies and applicants.

Reliability

NONREQ-8: The platform should have a high level of reliability, minimizing downtime or service interruptions.

NONREQ-9: Data integrity measures are in place to ensure accuracy, and regular data backups prevent the potential loss, contributing to a reliable platform.

NONREQ-10: High system availability is maintained through robust hosting solutions, redundancy for critical components, and security measures to protect against vulnerabilities and loss of user information.

Performance

NONREQ-11: The platform is optimized for low-latency response times, undergoing regular performance testing to identify and address any bottlenecks.

NONREQ-12: Scalability measures and load balancing techniques are implemented to handle increased user traffic efficiently and maintain optimal performance.

Supportability:

NONREQ-13: Provide documentation and training materials for administrators, companies, and applicants.

NONREQ-14: Offer customer support channels to address user queries and issues promptly.

NONREQ-15: Comprehensive documentation guides administrators, companies, and applicants, with additional troubleshooting guides and FAQs for user support.

NONREQ-16: Training sessions and ongoing support channels, such as email or chat, are provided, and a systematic approach to software updates ensures continuous improvement while minimizing disruptions.

Security

NONREQ-17: Data transmitted and stored should be encrypted to ensure the confidentiality and integrity of user information.

On-Screen Appearance Requirements:

Identifier	PW	Requirement
ONSREQ-1	4	Design an intuitive and responsive layout that adapts to different screen sizes and resolutions. Provide clear navigation paths, ensuring users can easily find and access relevant information.
ONSREQ-2	3	Define a set of consistent icons and symbols for actions, alerts, and other visual cues to enhance user understanding.
ONSREQ-3	3	The platform's visual design should align with the school's brand guidelines, ensuring a consistent and cohesive appearance.

FUNCTIONAL REQUIREMENTS SPECIFICATION

Stakeholders:

1. UB Students (Internal Users)

- Easily access the UB Job Portal.
- Share personal info for job applications.
- Search for jobs that match their skills.
- Communicate smoothly with potential employers.

2. Employers/Companies (External Users)

- Post job openings on the UB Job Portal.
- Check out applicant profiles.
- Communicate with applicants about job opportunities.
- Manage and review job applications.

3. Administrators

- Verify and manage user registrations.
- Ensure posted jobs meet eligibility criteria.
- Provide support to both applicants and employers.

Actors and Goals

Actor	Type	Goals
Applicant	Initiating	Create an account Upload, delete, and update personal information and documents Apply for jobs Message employers about a job if considered
Employer	Initiating	Post job applications Upload, delete, and update company information Accept potential candidates Message candidates about a job and communicate further steps in the application process.
Administrator	Initiating	See system usage information

Use cases

Name	Description	Requirement Covered
UC 1 - Registration	Allows the system to create a profile of a user using their personal or business information.	REQ-1 REQ-2 REQ-4 NONREQ-1 NONREQ-2 NONREQ-5 NONREQ-6 NONREQ-7 ONSREQ-3
UC 2 - Authentication	Ensures that ONLY UB students and legitimate businesses are allowed to register and login.	REQ-1 REQ-4 NONREQ-5 NONREQ-17 ONSREQ-3
UC 3 - FilterJob	To allow the system to decipher which listings are legitimate.	REQ-3 ONSREQ-1
UC 4 - ManageProfile	Allows system users to upload, edit, and delete their profile information.	REQ-2 NONREQ-1 NONREQ-2
UC 5 - PostJob	To allow employers to vacant job posts for applicants.	REQ-3 REQ-5 REQ-10 NONREQ-1 NONREQ-2 NONREQ-3 NONREQ-6 NONREQ-7 ONSREQ-3
UC 6 - SearchJobListing	To allow applicants to search through different job listings available.	REQ-5 REQ-6 NONREQ-4 NONREQ-7 ONSREQ-1
UC 7 - ApplyForJob	To allow applicants to apply for jobs by submitting their resume.	REQ-7 NONREQ-1 NONREQ-6 NONREQ-7 ONSREQ-1

		ONSREQ-2
UC 8 - Message	To allow both applicants and employers to communicate with each other via instant messaging.	REQ-9 NONREQ-1 ONSREQ-2
UC 9 - ViewStudentListing	Allows employers to view applicants personal information that they made public, as well their resumes.	REQ-8 NONREQ-7 ONSREQ-2
UC 10 - AnalyzeStatistics	To allow the administrator to view, provide reporting & analytics of usage statistics.	REQ-11 NONREQ-2

1. Student Registration and Profile Creation

- Description: Students sign up on the UB Job Portal and create profiles by sharing details about themselves, education, and skills.
- Actors: UB Student, System

2. Job Vacancy Posting by Employers

- Description: Employers upload job openings to the UB Job Portal, providing details like job descriptions, qualifications, and deadlines.
- Actors: Employer, System

3. Job Search and Application by Students

- Description: Students look for jobs, view details, and submit applications based on their preferences and qualifications.
- Actors: UB Student, System

4. Communication between Employers and Students

- Description: Employers and applicants chat through the platform regarding job applications, interview schedules, and application status.
- Actors: UB Student, Employer, System

5. Job Application Tracking by Students

- Description: Students track their job applications, receive updates, and efficiently manage multiple applications.
- Actors: UB Student, System

6. Job Application Review and Shortlisting by Employers

- Description: Employers review applications, shortlist candidates, and manage the recruitment process within the platform.
- Actors: Employer, System

Use Case Diagram

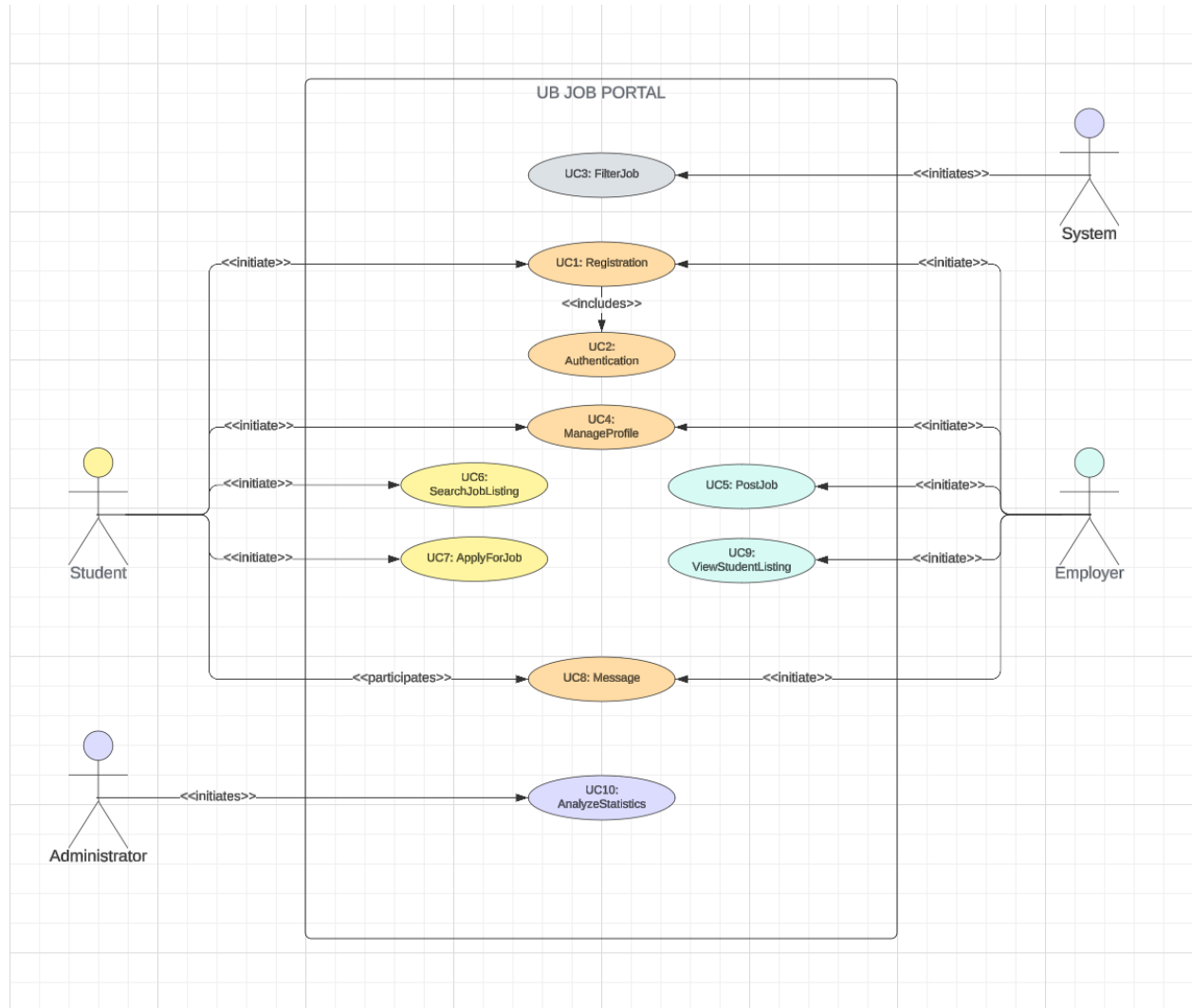


Figure 1 - Use Case Diagram of UB Job Portal

Traceability Matrix

PW Value		UC 1 - StudentRegistration	UC 2 - StudentAuthentication	UC 3 - EmployerRegistration	UC 4 - EmployerAuthentication	UC 5 - Filter Job	UC 6 - Insert-Info	UC 7 - ChangeStudentInfo	UC 8 - UploadCompanyInfo	UC 9 - ChangeCompanyInfo	UC 10 - CreateJob	UC 11 - SearchJobListing	UC 12 - ApplyJob	UC 13 - Message	UC 14 - ViewDocuments	UC 15 - ViewPersonalInfo	UC 16 - Verify	UC 17 - AnalyzeStatistics
5	REQ-1	X		X														
2	REQ-2	X	X	X	X													
1	REQ-3						X											
2	REQ-4							X										
3	REQ-5																	
5	REQ-6								X		X							
1	REQ-7									X								
5	REQ-8					X						X						
2	REQ-9					X						X						
1	REQ-11												X					
1	REQ-12														X	X		
2	REQ-13													X				
3	REQ-14													X				
1	REQ-15		X		X												X	
3	REQ-16																	X
2	NOREQ-1	X		X		X	X	X	X	X								
4	NOREQ-2							X	X	X								
5	NOREQ-3																	
2	NOREQ-4																	
3	NOREQ-5																	
5	NOREQ-6	X		X														X
2	NOREQ-7	X		X		X	X	X	X	X								
2	NOREQ-8																	
4	ONSREQ-1	X		X			X	X	X	X		X						
3	ONSREQ-2	X		X														
3	ONSREQ-3	X		X														
Total Weight		26	3	26	3	11	9	14	17	13	5	11	2	5	1	1	1	8

Table 1. Traceability Matrix according to system requirements and use cases

Fully-Dressed Description

UC-1	Registration
Related Requirements	REQ-1, REQ-2, NONREQ-1, NONREQ-6, NONREQ-7
Initiating Actor	Student
Actor's Goal	To allow the system to capture all fields entered when applicants register
Participating Actors	Administrator
Pre-Condition	
Post-Condition	
Flow of Events for Main Success Scenario	<p>→ 1. The applicant accesses the registration site via the web portal or application</p> <p>→ 2. They go to the sign up section of the portal to create a new account</p> <p>→ 3. The applicant inputs all necessary information on the signup page (eg. name, phone number, etc.)</p> <p>→ 4. Once done, the applicant will click the register button.</p> <p>← 5. The system will ask the applicant to confirm the registration. The applicant confirms.</p> <p>← 6. The system updates itself with the new information added by the applicant</p>

UC-5	PostJob
Related Requirements	REQ-3, REQ-5,REQ-10, NONREQ-3
Initiating Actor	Employer
Actor's Goal	Create a post with details of an available job for an applicant to view.
Participating Actors	
Pre-Condition	The employer is registered and have logged in.
Post-Condition	The job post is added to a list of other posts called a job listing. Applicants can then view this job listing.
Flow of Events for Main Success Scenario	→ 1. The employer clicks a button to add a new job post. ← 2. The system displays a form to get the details of the new post. → 3. (a) The employer enters the job details. (b) The employer submits the form. ← 4. (a) The system adds the new post details into the database. (b) The system displays that new post in the applicants' feeds.

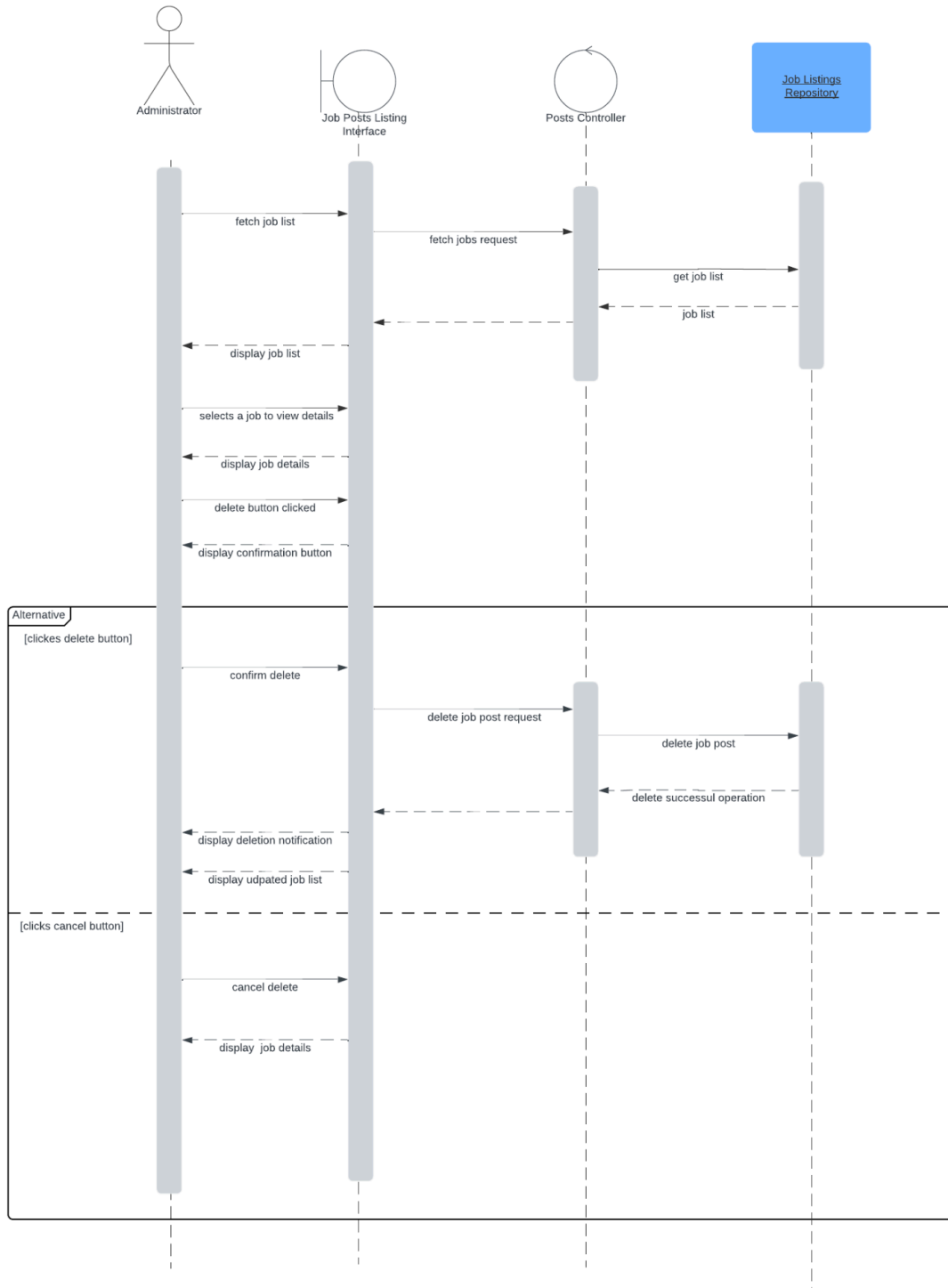
UC-2	Authentication
Related Requirements	REQ-1, REQ-4, NONREQ-5, NONREQ-17
Initiating Actor	System
Actor's Goal	Ensure that applicants are UB students and employers are verified and trusted.
Participating Actors	Administrator
Pre-Condition	The administrator is logged in and can view a list of all users.
Post-Condition	The administrator can disable the account of an employer or applicant.
Flow of Events for Main Success Scenario	<p>Admin checks an employer's account → 1. The admin selects a button to view all employers. ← 2. The system returns the list of employers → 3. The admin selects to view the employer's details. ← 4. The system displays the employer's details → 5. (a) The admin selects to disable/suspend the employer's account. (b) The admin selects and submits the reason for the suspension. ← 6. (a) The system updates the employer's account in the database, not allowing logins for that account. (b) The system displays to the admin that the account is disabled.</p>
Flow of Events for Extensions(Alternate Scenarios)	<p>Admin checks an applicant's account → 1. The admin selects a button to view all applicants. ← 2. The system returns the list of applicants → 3. The admin selects to view the applicant's details. ← 4. The system displays the applicant's details → 5. (a) The admin selects to disable/suspend the applicant's account. (b) The admin selects and submits the reason for the suspension. ← 6. (a) The system updates the applicant's account in the database, not allowing logins for that account. (b) The system displays to the admin that the account is disabled.</p>

UC-7	ApplyForJob
Related Requirements	REQ-7, NONREQ-1, NONREQ-6, NONREQ-7, ONSREQ-1, ONSREQ-2,
Initiating Actor	Applicant
Actor's Goal	To forward an applicant's information to an employer regarding a job post.
Participating Actors	
Pre-Condition	The applicant is logged in and shown a list of posted jobs. The applicant has already uploaded their resumes to their profile.
Post-Condition	The employer is notified of the application and receives the applicant's information.
Flow of Events for Main Success Scenario	→ 1. The applicant selects a button to apply for the job. ← 2. The system prompts to confirm the option. → 3. The applicant confirms the action. ← 4. (a)The system retrieves the applicants' information from a database, along with their resume documents. (b) the system creates this user's application. (c) The system sends the application to the employer.

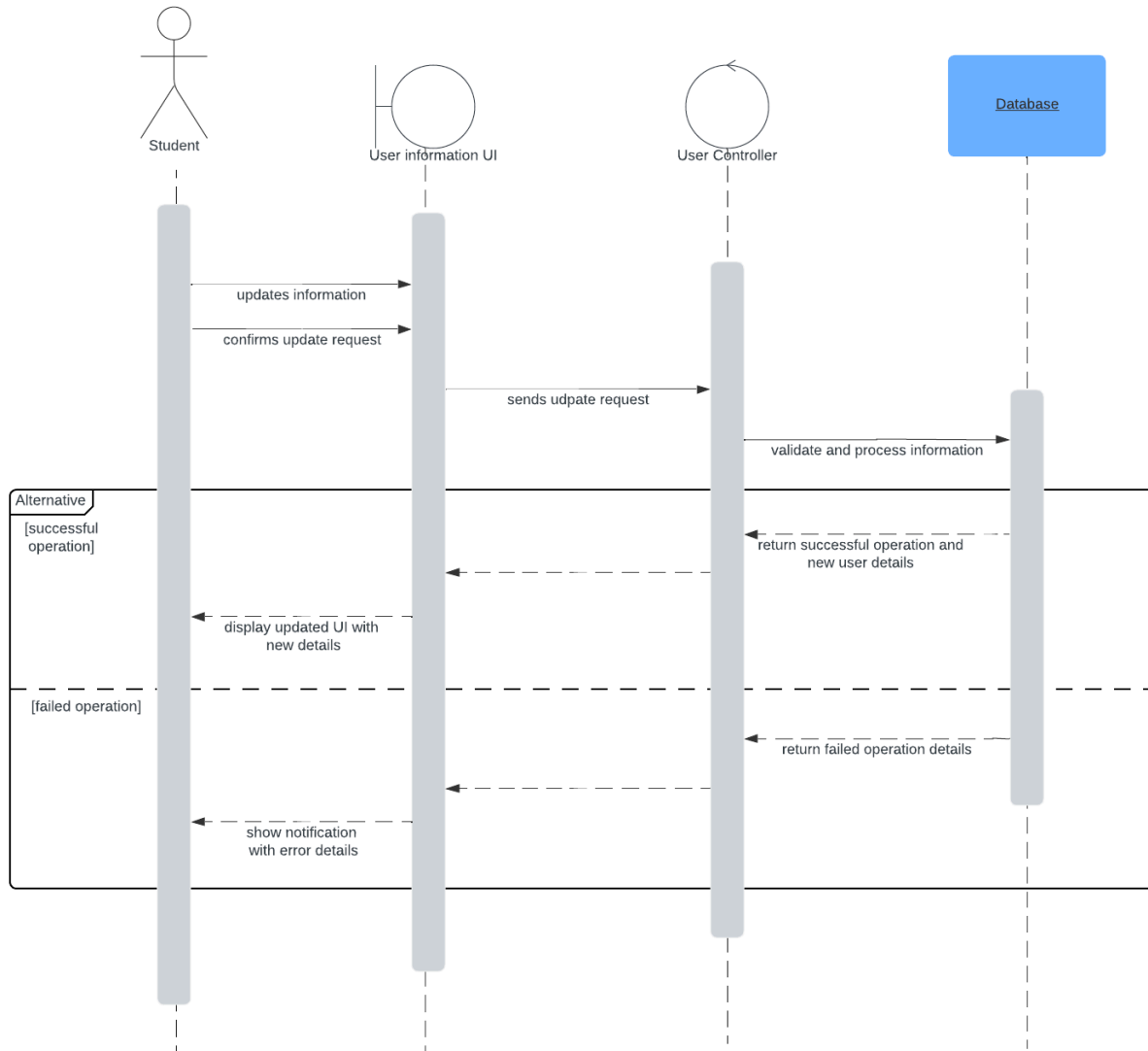
UC-6	SearchJobListing
Related Requirements	REQ-5, REQ-6, NONREQ-4, NONREQ-7, ONSREQ-1
Initiating Actor	Applicant
Actor's Goal	To search and find different jobs based on different criterias.
Participating Actors	
Pre-Condition	The applicant is logged in, and there are multiple job posts.
Post-Condition	A list of job posts that fit the criteria is displayed the applicant applies for those job posts.
Flow of Events for Main Success Scenario	→ 1. The applicant selects a button to see the search criteria options. ← 2. The system displays the options. → 3. The user selects one or multiple filter/search options. ← 4. (a) The system generates a query from the search options. (b) The system queries the database for related job posts.(c) The system displays the job posts to the applicants.

Sequence Diagrams

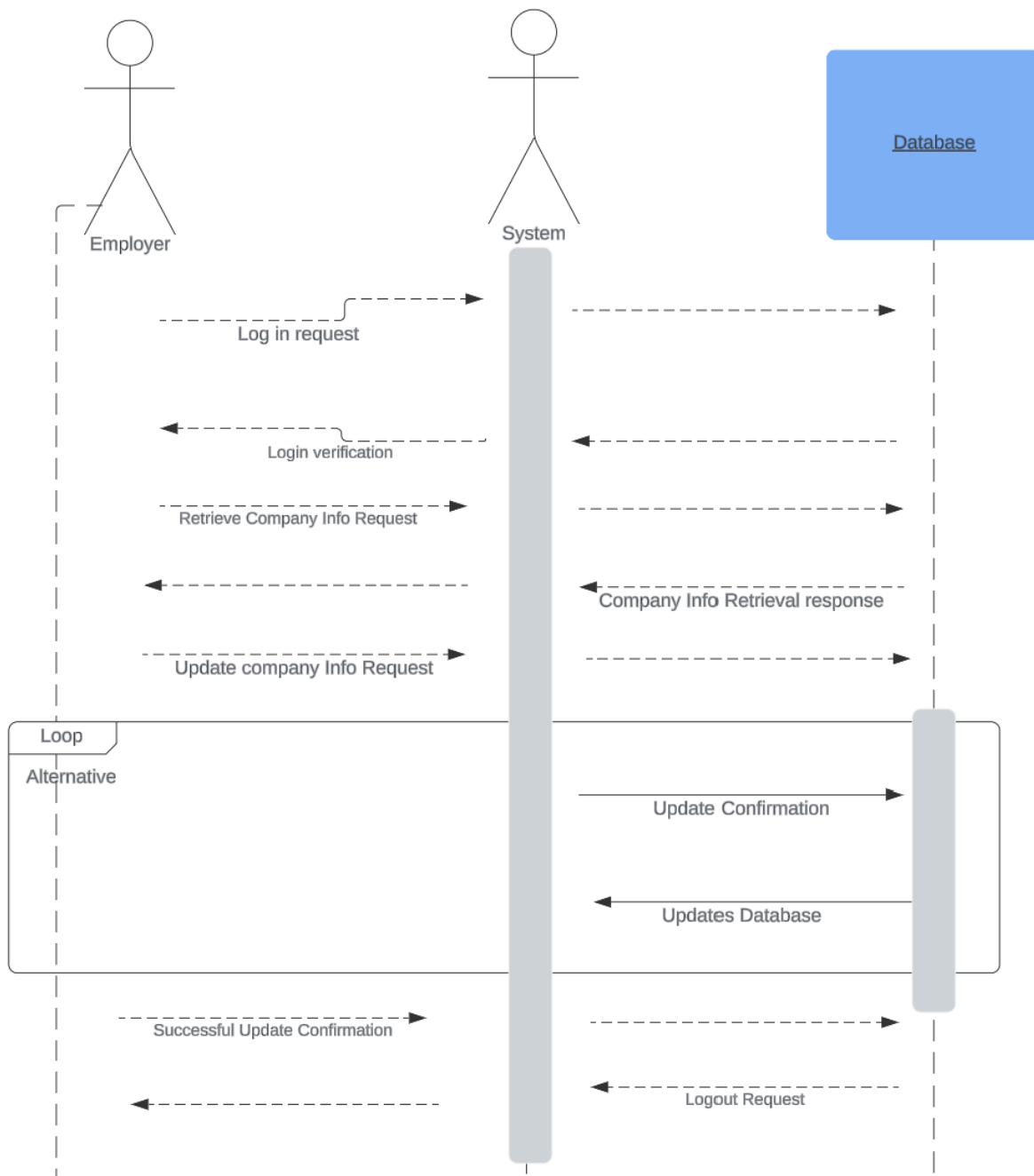
UC5 - Filter Job



UC8 Upload Company Information



UC9 ChangeCompanyInfo



Interface Specification

Preliminary Design

UC 1 - StudentRegistration

To create an account to apply for jobs, an applicant inputs their first name, last name, email address, phone number, and password. With this account, applicants can be able to then log in using their email and password. By creating an account, applicants have access to job listings and have access to apply for jobs.

University of Botswana
UB
UB Job Portal

Find Jobs, Apply, Get Hired! Join Now!

First name Last name

Email Address

Phone number Field of study

Password

Confirm Password

[Continue](#)

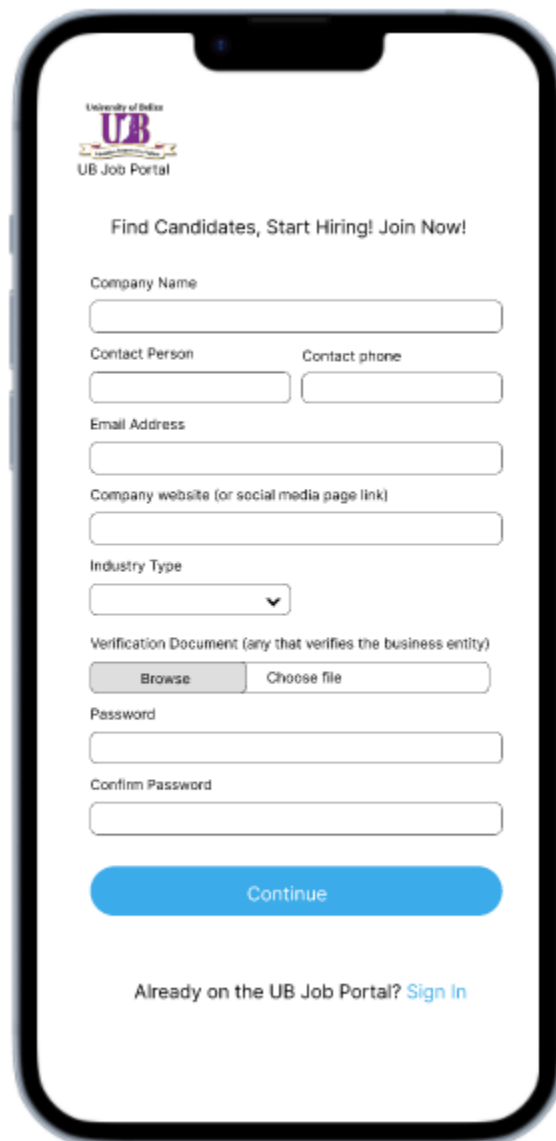
Already on the UB Job Portal? [Sign In](#)

Want to be a registered employer ? [Create an account](#)

Figure 2 - Student registration screen

UC 3 - EmployerRegistration

To create an account to post jobs, their identity as an official business needs to be confirmed by an administrator. An employer inputs the following information to the sign-up form: company name, a contact person's name, and phone number, email address, company/social media website link, industry type, and any file that helps to verify their business entity. After approval, an employer can log in using their email and password. With this account, employers can create job posts and handle candidates.



The image shows a smartphone screen displaying the 'UB Job Portal' registration form. At the top, the 'University of Belize' logo and 'UB Job Portal' text are visible. Below this is a heading 'Find Candidates, Start Hiring! Join Now!'. The form contains several input fields: 'Company Name', 'Contact Person', 'Contact phone', 'Email Address', 'Company website (or social media page link)', 'Industry Type' (a dropdown menu), 'Verification Document (any that verifies the business entity)' (with 'Browse' and 'Choose file' buttons), 'Password', and 'Confirm Password'. A blue 'Continue' button is at the bottom of the form. Below the button, there is a link: 'Already on the UB Job Portal? Sign In'.

University of Belize
UB
UB Job Portal

Find Candidates, Start Hiring! Join Now!

Company Name

Contact Person Contact phone

Email Address

Company website (or social media page link)

Industry Type

Verification Document (any that verifies the business entity)

Password

Confirm Password

Already on the UB Job Portal? [Sign In](#)

Figure 3 - Employer registration screen

UC 5 - FilterJob

In order for the administrator to verify the credibility of job listings he/she will cycle through each job vacancy that was posted by companies and the administrator has the option to select the specific one that didn't meet the criteria and it can be deleted with the tap of the “delete button”, administrator will be prompted with a warning message indicating that it won't be recovered after deletion..

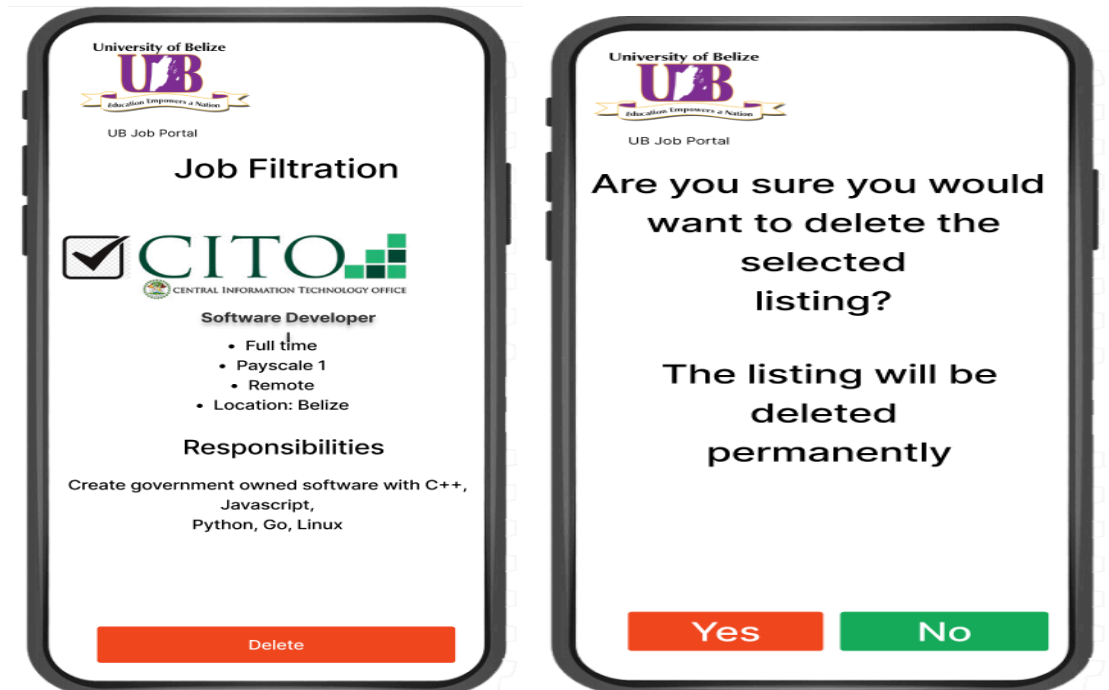


Figure 4 - Job Filtration Screen

UC9 ChangeCompanyInfo

To update the information so employers can efficiently manage and edit job postings with the latest company information for potential candidates.

The image shows a mobile application interface for updating company information. At the top, the University of Belize logo and 'UB Job Portal' are displayed. The main heading is 'Update Company Information'. Below this, there are several input fields: 'Company Name' with a placeholder 'Current Company Name', 'Contact Person' with 'Current Contact Person', 'Contact Phone' with 'Current Contact Phone', 'Email Address' with 'Current Email Address', 'Company website (or social media page link)' with 'Current Company Website', and 'Industry Type' with 'Current Industry Type'. A 'Verification Document (any that verified the business entity)' section contains a file input field with 'CurrentFile.pdf' and an 'Edit File' link. At the bottom, there are two buttons: 'Save Changes' (blue) and 'Cancel' (red).

University of Belize
UB
The Quality Standard of Excellence
UB Job Portal

Update Company Information

Company Name
Current Company Name

Contact Person
Current Contact Person

Contact Phone
Current Contact Phone

Email Address
Current Email Address

Company website (or social media page link)
Current Company Website

Industry Type
Current Industry Type

Verification Document (any that verified the business entity)
CurrentFile.pdf [Edit File](#)

[Save Changes](#) [Cancel](#)

Figure 5 - Update Company Information Screen

User Effort Estimation

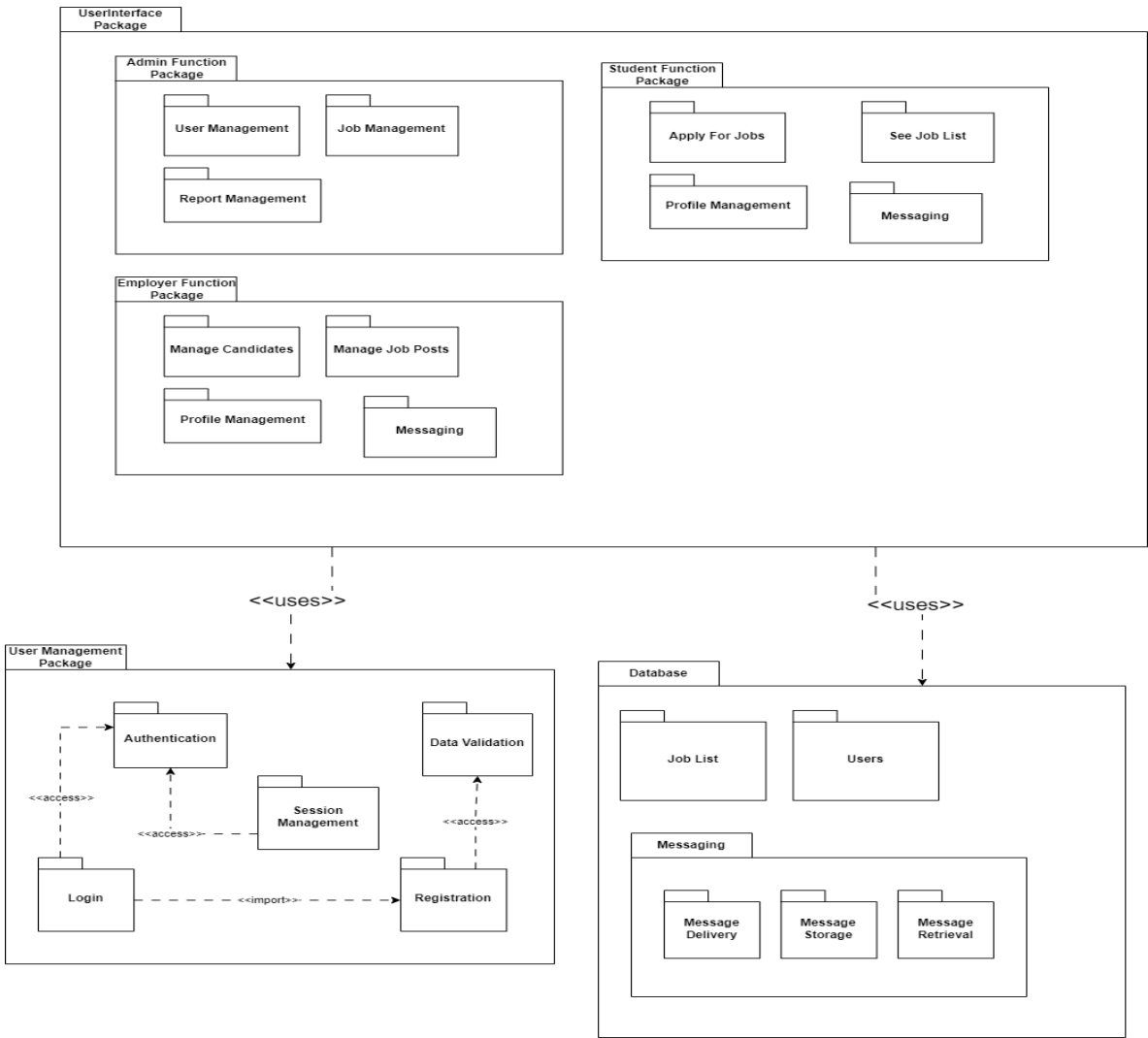
Scenario	Navigation	Data Entry
Student Registration	Click on first name field Click on last name field Click on email address field Click on phone number field Click on field of study dropdown Click field of study from drop-down list Click on password field Click on confirm password field Click continue button	Input first name Input last name Input email address Input phone number Input password Reinput password
Employer Registration	Click on company name field Click on contact person field Click on contact phone field Click on email address field Click on company website field Click on industry type dropdown Click industry type from the drop-down list Click on file input for verification document Click file input selection and confirm button Click on password field Click on confirm password field Click continue button	Input company name Input contact person Input contact phone Input email address Input company website Input password Reinput password
Update Company Info	Click on company name field Click on contact person field Click on contact phone field Click on email address field Click on company website field Click on industry type dropdown Click on file input for verification document Click on Save Changes button	Edit company name Edit contact person Edit contact phone Edit email address Edit company website Edit Industry type Edit file for verification document
Filter Job	Click on invalid job vacancy Click on the delete button	None

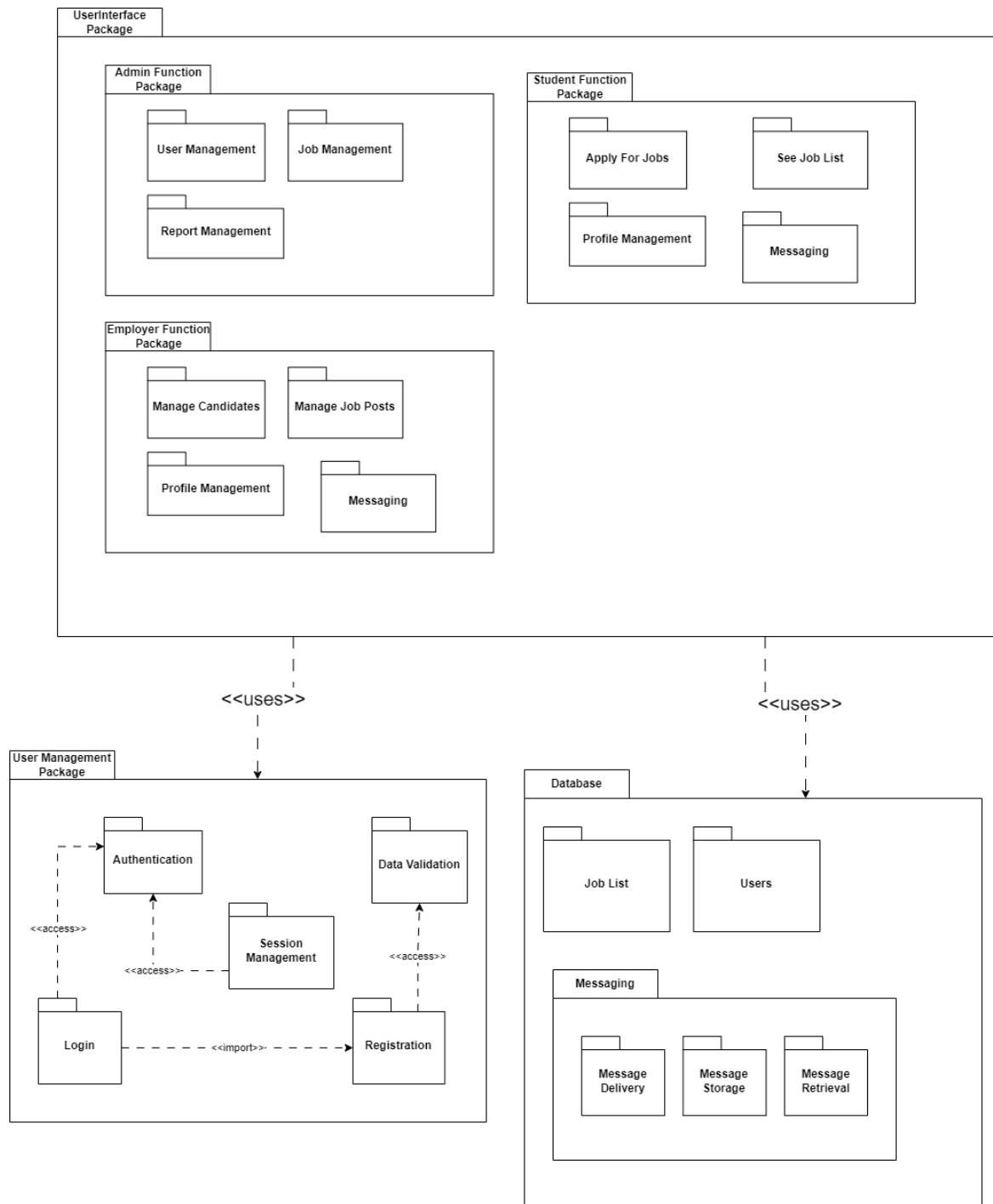
	Prompt window appears Click on yes to proceed with deletion Click on no to keep the job vacancy	
--	---	--

System Architecture

Identifying Subsystems

UML Package Diagram





Architecture Styles

The architecture of the job portal system follows a client-server model, with a web-based user interface for employers, applicants, and administrators interacting with a backend server for data processing and storage. Therefore, employers and applicants can access the service from any device with an internet connection and a web browser. Administrators may view reports and usage statistics on any device, but a desktop computer is most suitable. The UB job portal must handle personal user information over the internet, so reliable protocols such as HTTPS is mandatory.

Database

The system will utilize MySQL as the main relational database technology for all its storage, information processing, and retrieval needs.

Data Communication

The system will utilize a RESTful API so that any authorized requesting system can access and manipulate web resources on the UB Job Portal. All requests are performed using the internet's HyperText Transfer Protocol, and are restricted to GET, PUT, POST, and DELETE requests.

Object Oriented

The system will use object-oriented architecture and design principles to take advantage of modularity, reusability, improved maintainability, and enhanced collaboration.

Event Driven

Our system will use event-driven architecture as users perform actions such as applying for jobs, searching, filtering, and handling personal information. The system must log these events as usage reports are fundamental to the administrator and developers. These events also need to be processed promptly to ensure a good user experience.

Asynchronous messaging

Real-Time Communication: Asynchronous messaging can enable real-time communication between users, such as employers and job seekers, allowing immediate responses and quicker interactions during the hiring process.

Notification System: Asynchronous messaging can be used to implement a notification system that informs users about important updates, such as new job postings, application status changes, or messages from employers/recruiters.

Mapping Subsystems to Hardware:

The system primarily operates as a web application, with the client-side subsystem running on users' devices such as laptops, desktops, or mobile phones, utilizing standard web browsers. The server-side subsystem, including the database management system, runs on a centralized server. This server can be hosted either on-premises or on a cloud platform such as Amazon Web Services (AWS) or Microsoft Azure. Communication between the client and server subsystems occurs over the internet, utilizing standard network protocols such as HTTP or HTTPS. Additionally, data transfer between the server and the database requires a stable network connection with sufficient bandwidth to handle concurrent user requests efficiently.

Connectors and Network Protocols:

To enable companies to hire University of Belize students via a web portal, use HTTPS for secure communication, OAuth 2.0 for user authentication, and ODBC/JDBC for database connectivity. Employ WebSocket for real-time updates, SMTP for emails, and RESTful APIs or GraphQL for integration. Implement TLS/SSL for data security, strong measures against unauthorized access, and consider MQTT for instant messaging.

Global Control Flow

Execution Orderliness:

The system adheres to a systematic, procedure-driven approach, ensuring a uniform user experience through a predefined sequence of steps.

It maintains a structured process, wherein each user undergoes the same linear progression when engaging with employers, submitting resumes, and participating in chats.

Time Dependency:

Notably, the system abstains from the use of timers, devoid of temporal constraints.

The system operates as a non-real-time entity, functioning within an event-response framework, wherein user actions prompt responses without adherence to real-time intervals.

Event-Response Type:

The system is architected as an event-response model, characterized by its responsiveness to user-initiated actions absent explicit real-time considerations.

It abstains from temporal periodicity, allowing for the natural unfolding of events as users engage in interactions, devoid of predefined time constraints.

Hardware Requirements:

The hardware requirements will likely depend on many factors, such as the scale of the portal, the number of users it needs to support, and different features that it offers. These are some of the basic hardware requirements

Storage → A sufficient amount of storage to store all the data in the portal. A SSD would be a faster choice.

RAM (memory) → Minimum of at least 2GB of RAM just for the smooth operation of request.

Processor (CPU) → A good CPU (doesn't have to be the latest version) to handle the load of the portal.

Plan of Work

1. Requirements Gathering:

- Collaborate with stakeholders to understand their needs and expectations.
- Identify essential features, user roles, and specific integration requirements.

2. System Architecture Design:

- Plan the overall structure of the web portal, including database design, server architecture, and technology stack.
- Consider scalability, security measures, and performance optimization.

3. Technology Stack Selection:

- Choose appropriate programming languages, frameworks, and libraries based on project requirements.
- Select a database management system that aligns with scalability and data storage needs.

4. User Interface (UI) and User Experience (UX) Design:

- Design an intuitive and user-friendly interface that meets the needs of both companies and applicants.
- Implement responsive design for a seamless experience across devices.

5. Backend Development:

- Develop server-side logic and functionality to handle user authentication, data processing, and integration with external services.
- Implement RESTful APIs or GraphQL for smooth communication between the front and back end.

6. Frontend Development:

- Create the user interface based on the design, ensuring a responsive and visually appealing portal.
- Integrate frontend components with the backend using chosen web protocols.

7. Database Implementation:

- Set up the database structure based on the design specifications.
- Implement necessary security measures to protect sensitive information.

8. Security Implementation:

- Integrate TLS/SSL for secure data transmission.
- Implement user authentication and authorization mechanisms, utilizing OAuth 2.0 or JWT.
- Employ encryption and secure coding practices to safeguard against potential vulnerabilities.

9. Integration with University Systems:

- Collaborate with the University of Belize's IT department to integrate the portal with existing university systems.
- Ensure compliance with university data policies and standards.

10. Testing:

- Conduct thorough testing, including unit testing, integration testing, and user acceptance testing.
- Identify and resolve bugs, usability issues, and performance bottlenecks.

11. Deployment:

- Deploy the web portal on a secure and scalable hosting environment.
- Implement necessary monitoring tools to ensure system stability.

12. Training and Documentation:

- Provide training sessions for users and administrators.
- Develop comprehensive documentation for ongoing maintenance and future updates.

13. Launch:

- Officially launch the web portal for companies to start hiring University of Belize students.
- Monitor performance and user feedback post-launch for further improvements.

14. Finalize for Final Demonstration:

- Prepare system for final demonstration in class

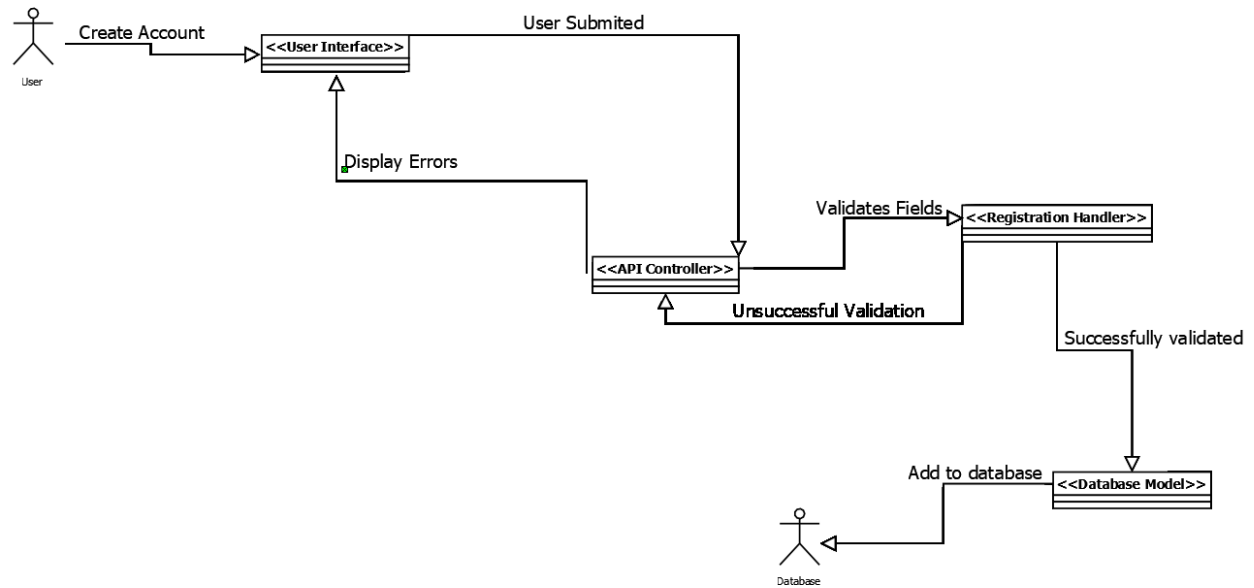
Gantt Chart for the development of the System:

Task	Timeline													
	March							April						
	1	8	10	16	21	27	31	7	14	17	19	22	26	28
Requirements Gathering	1 Day													
System Architecture Design		7 Days												
Technology Stack Selection			2 Days											
User Interface (UI) and User Experience (UX) Design				6 Days										
Backend Development					5 Days									
Frontend Development						6 Days								
Database Implementation							4 Days							
Security Implementation								7 Days						
Integration with University Systems									7 Days					
Testing										3 Days				
Deployment											2 Days			

[illegible]

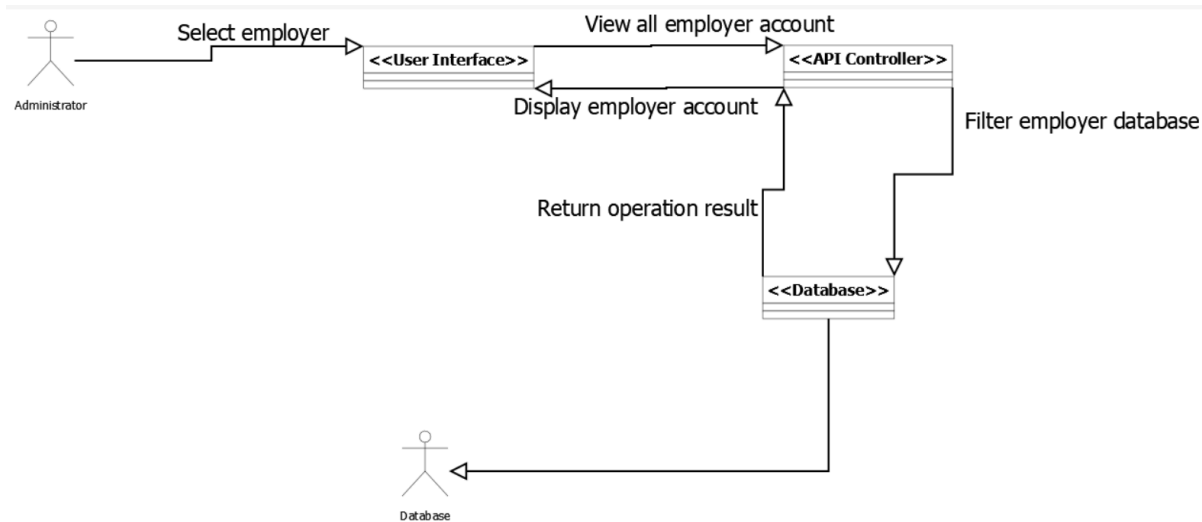
Analysis and Domain Modeling

Conceptual Model

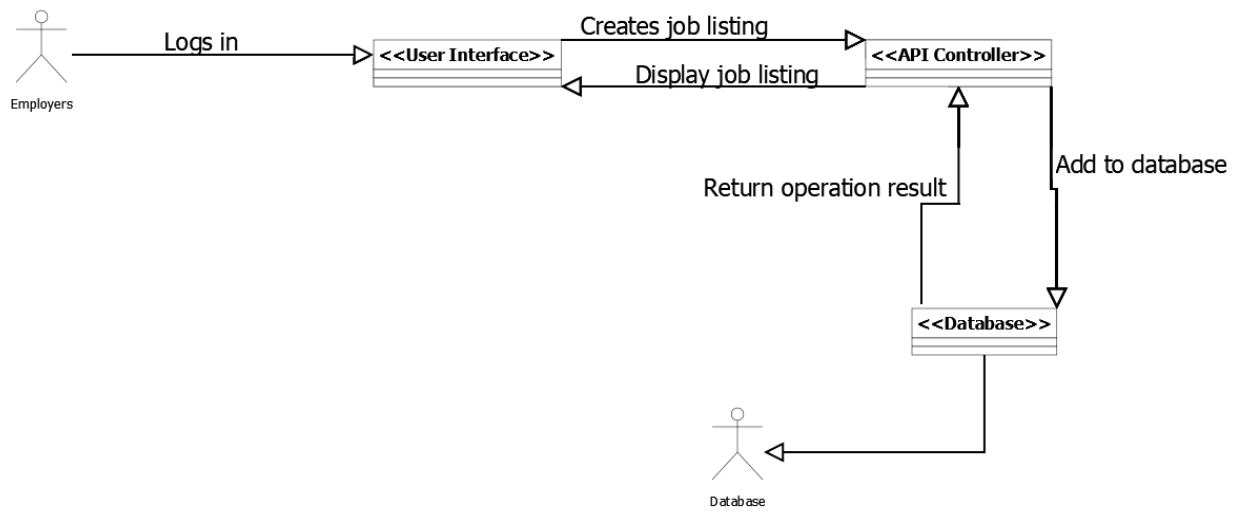


UC-1 Registration

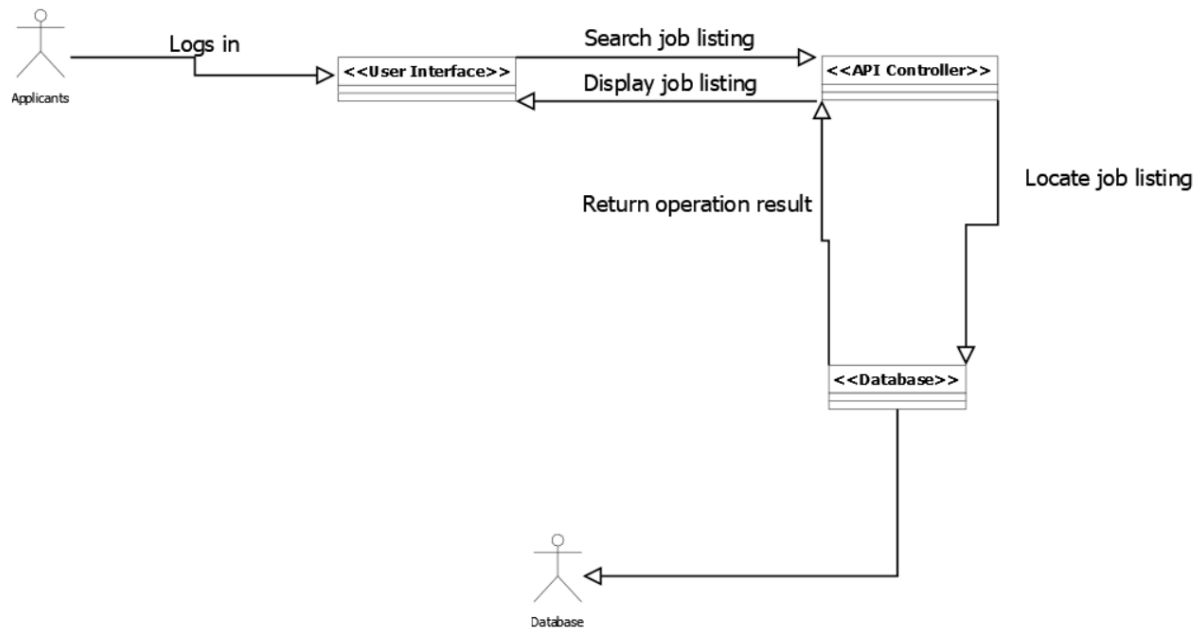
=



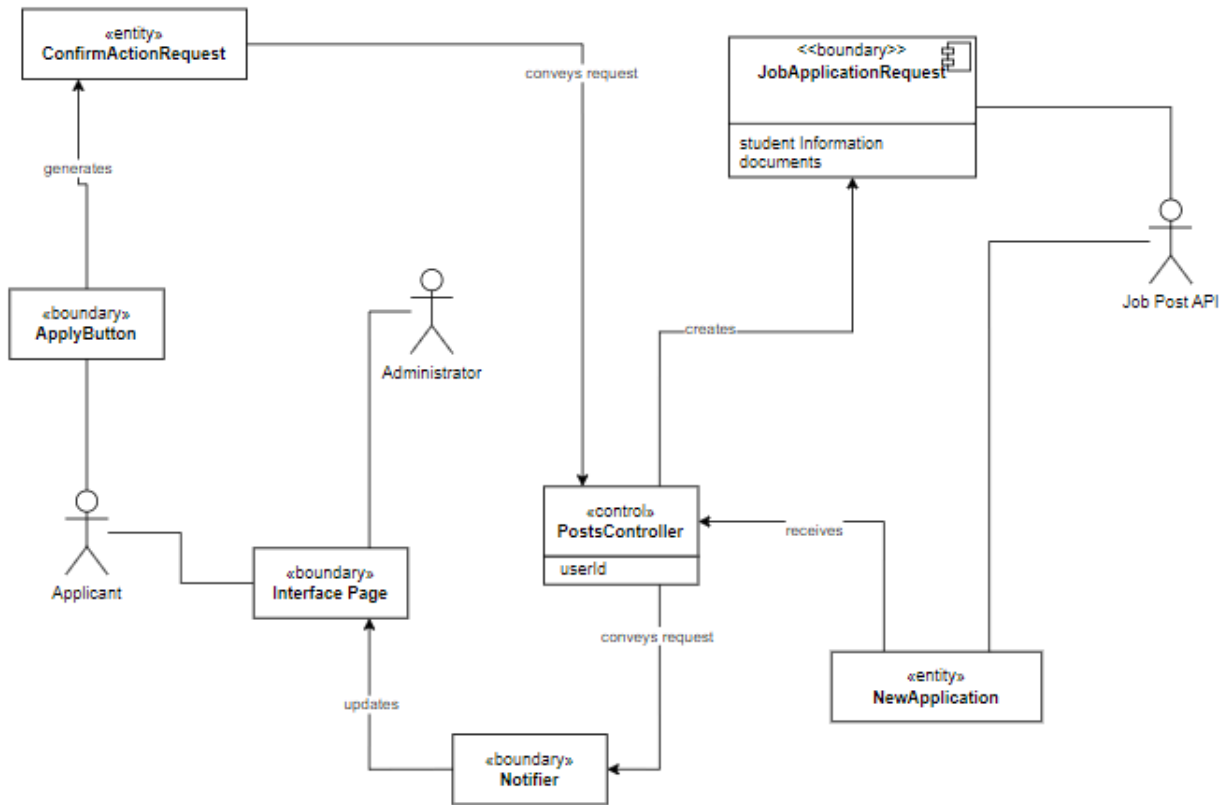
UC-2 Authentication



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:

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

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

System Operation Contracts

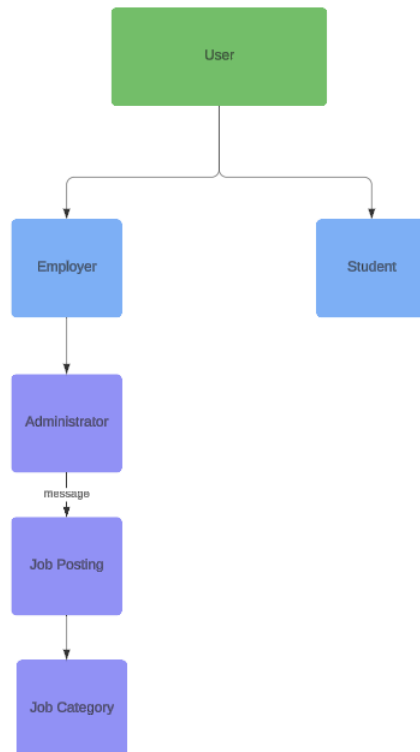
Operation	StudentRegistration()
Cross References	UC 1 - Student Registration
Preconditions	<ul style="list-style-type: none">• Students access the registration page through the web portal or app.• The registration page is shown
Postconditions	<ul style="list-style-type: none">• StudentRegistration() 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	<ul style="list-style-type: none">• Employers access the registration page through the web portal or app• The registration page is shown
Postconditions	<ul style="list-style-type: none">• EmployerRegistration() is called• Function checks if the employer is not already in the system• Once completed, the employer registration details wi updated into the database• Notifier calls to print update on the screen

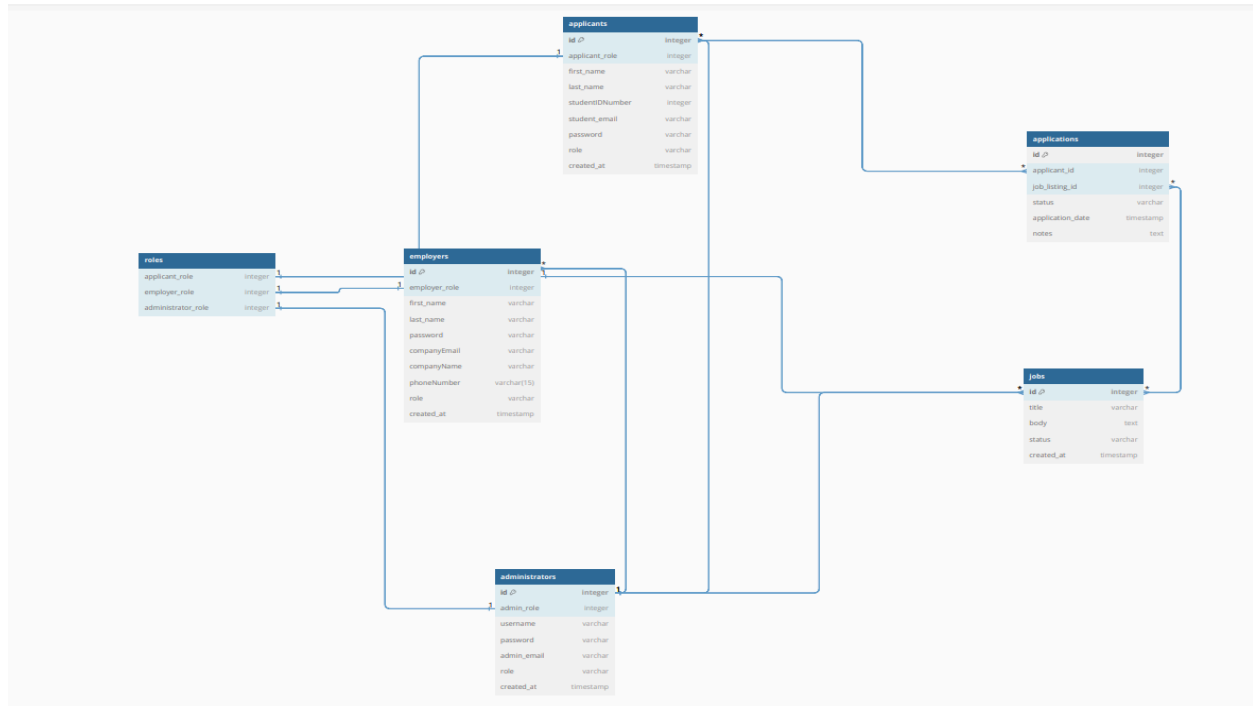
Operation	FilterJob()
Cross References	UC 5 - Filter Job
Preconditions	<ul style="list-style-type: none"> • Employer goes to the web page or app and logs in • The employer goes to the filter job section
Postconditions	<ul style="list-style-type: none"> • FilterJob() is called • Function should show a list of all the jobs that have been filtered to meet the specific requirements • If the filter gets changed, then the job listings should be updated

Operation	UpdateCompanyInfo()
Cross References	UC 8 - Update Company Info
Preconditions	<ul style="list-style-type: none"> • Employer goes into the web page or app and logs in • The page is presented • Update company information is shown
Postconditions	<ul style="list-style-type: none"> • UpdateCompanyInfo() 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	<ul style="list-style-type: none"> • Employer goes into the web page or app and logs in • The page is presented • Change Company Information is shown
Postconditions	<ul style="list-style-type: none"> • 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.



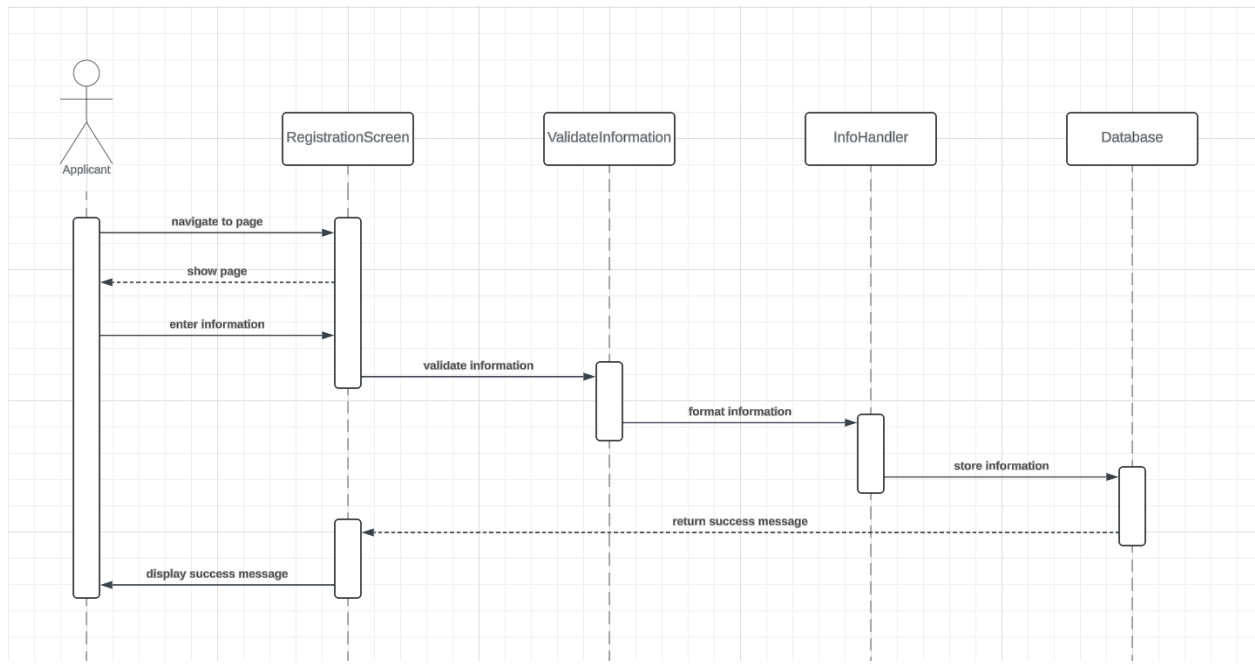
Data Models & Persistent Data Storage



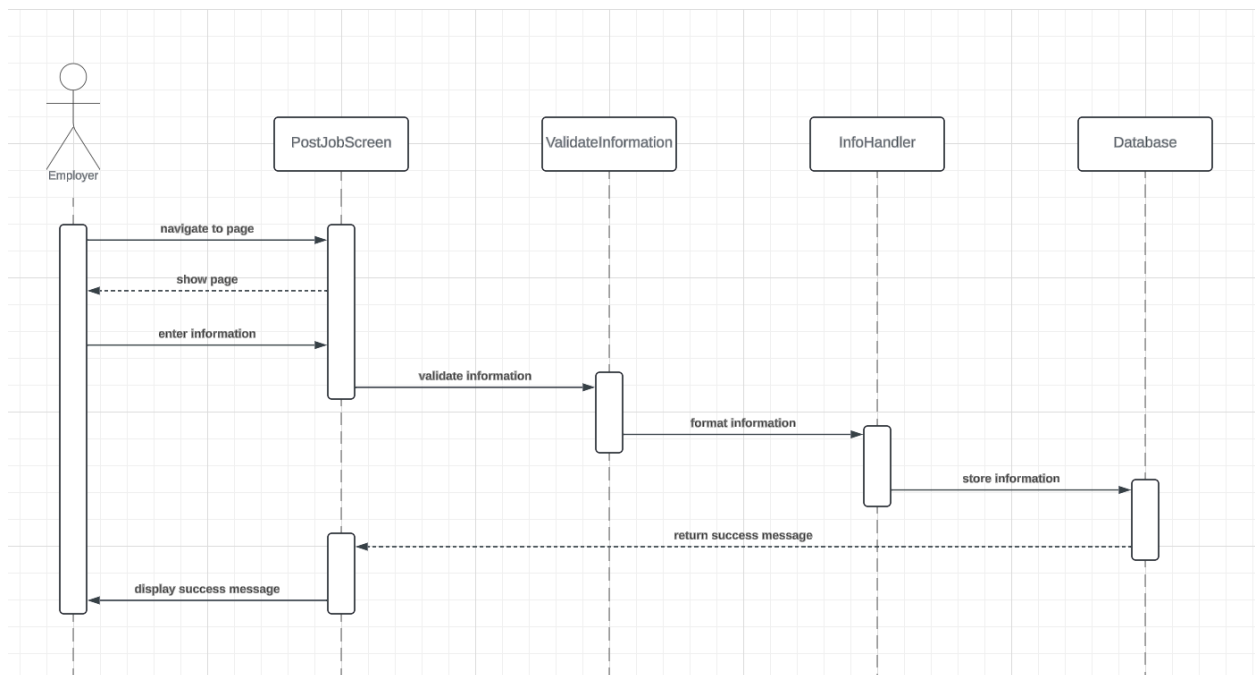
For this project, we do need persistent data storage. The data we need to have stored will be the login information from users and the job listings created by employers. We will do this by storing them in our database "Job Portal". We will also need Admin and User data to have their own tables. We will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

Interaction Diagrams

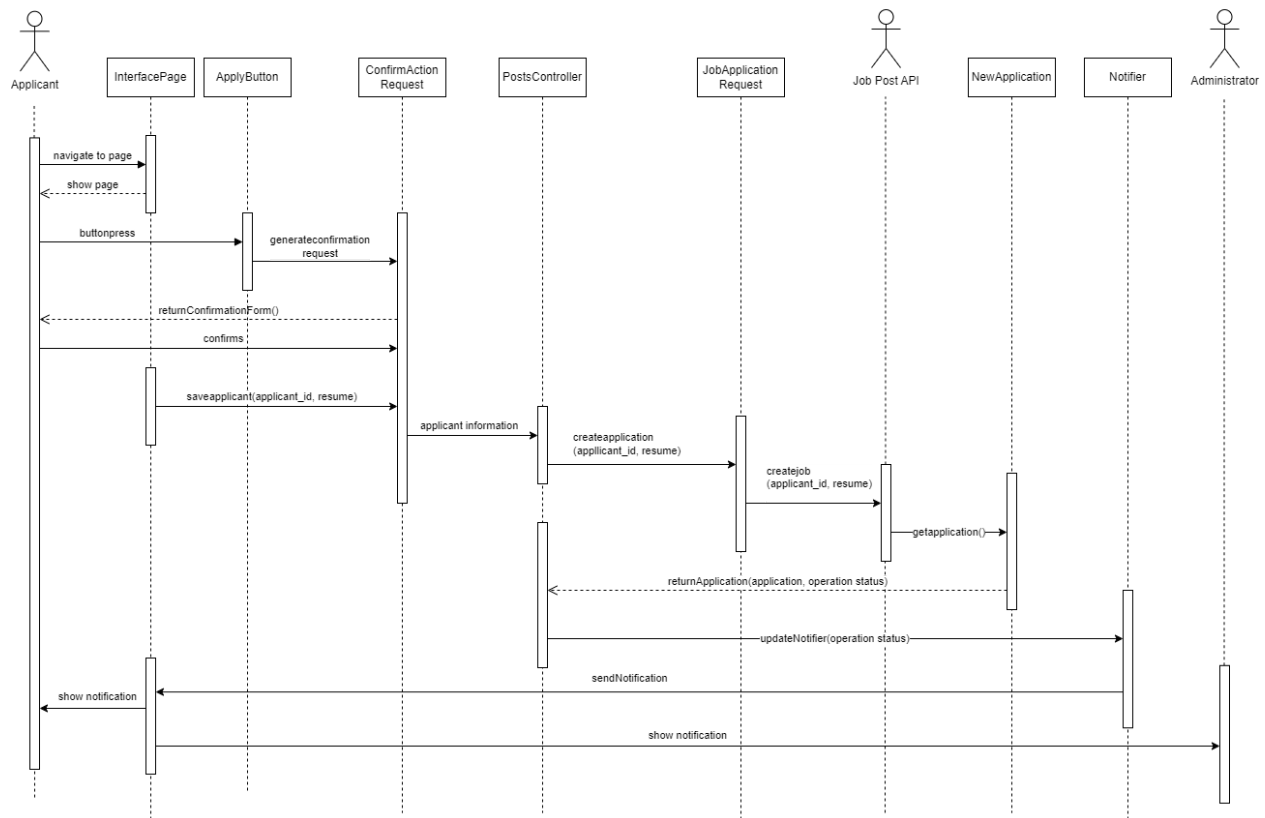
UC-1 Registration



UC-5 PostJob



UC-7 ApplyForJob

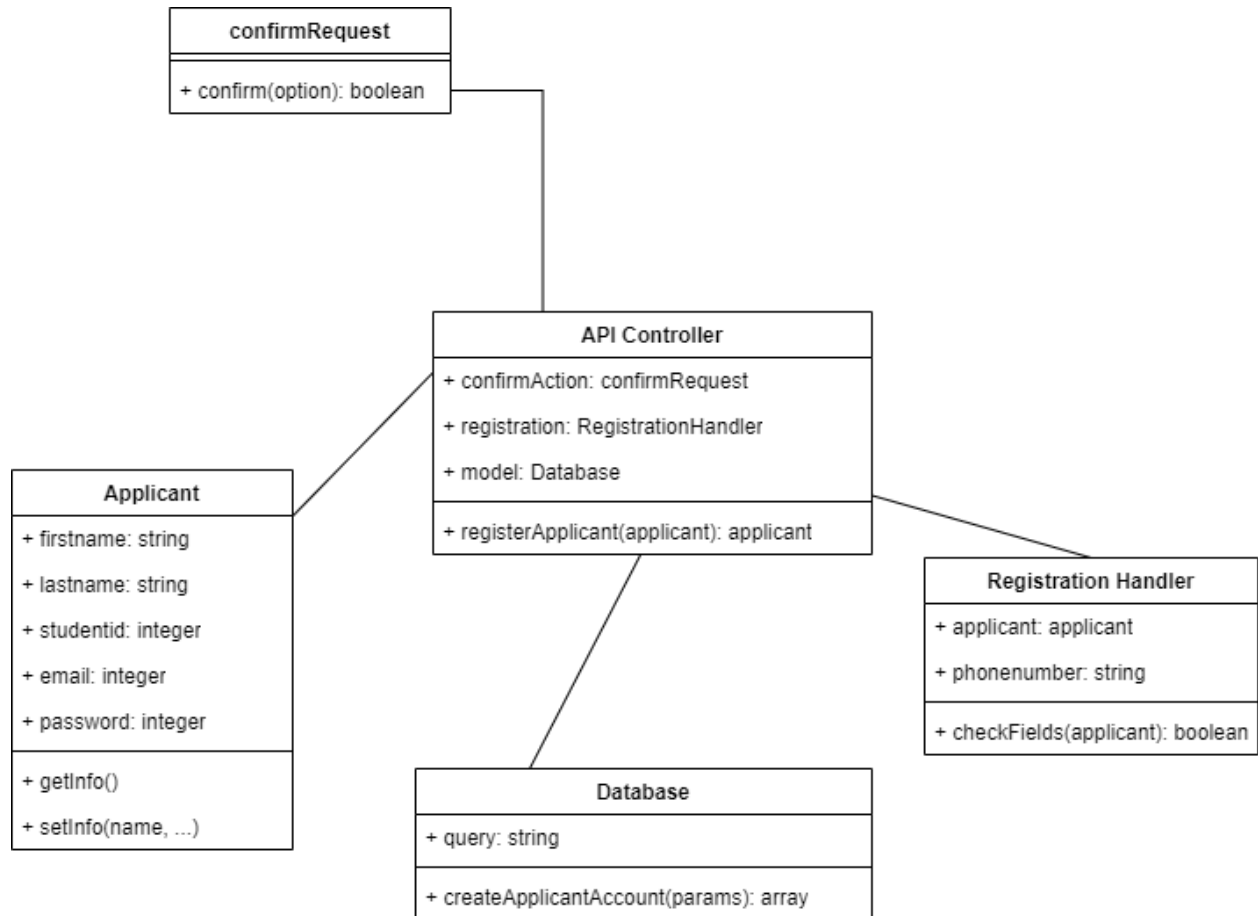


The applicant navigates to a URL, and the system returns the appropriate page. By clicking a button, the applicant generates a confirmationRequest which returns a form for a yes or no input. The user inputs a value of yes or no in a boolean representation, which is saved in the confirmationRequest. Upon 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.

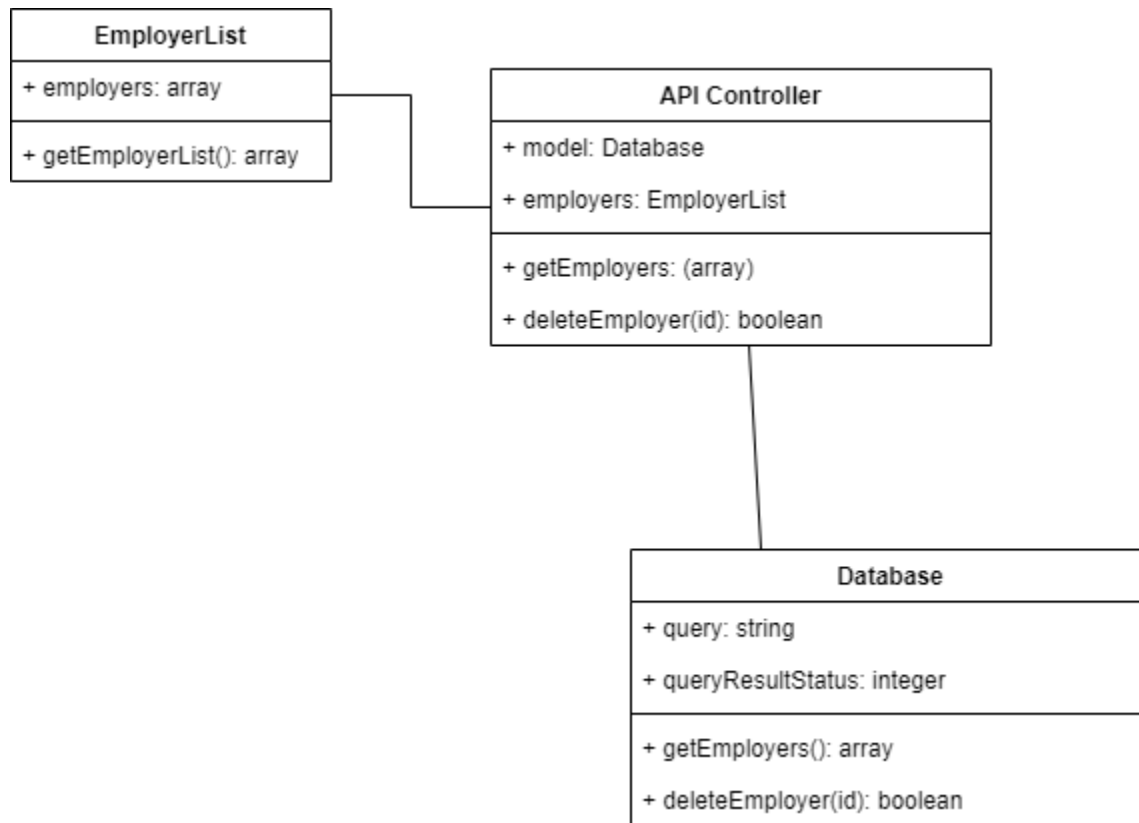
Class Diagram and Interface Specifications

Class Diagrams

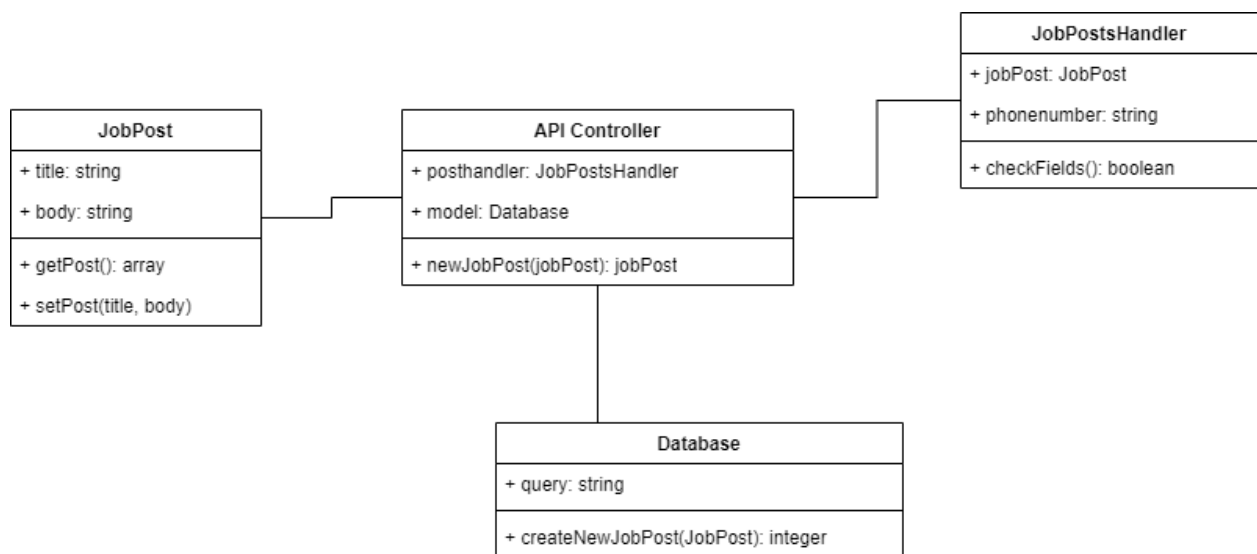
UC 1 - Registration



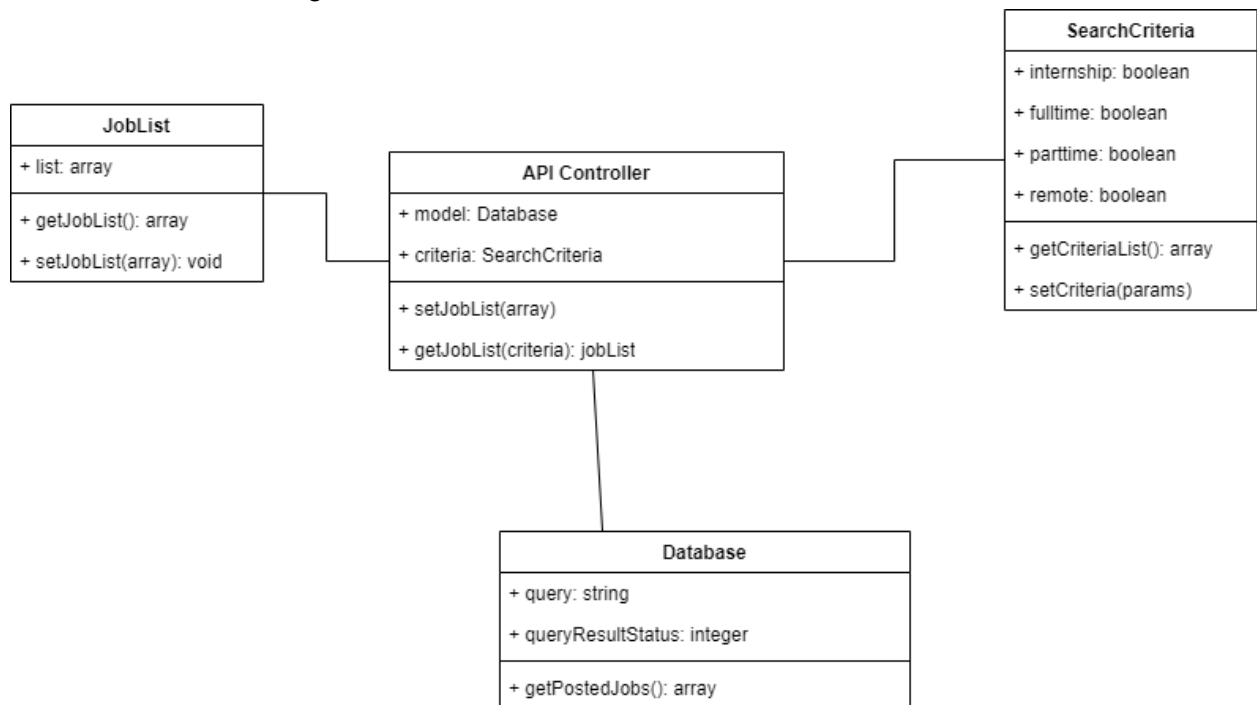
UC 2 - Authentication



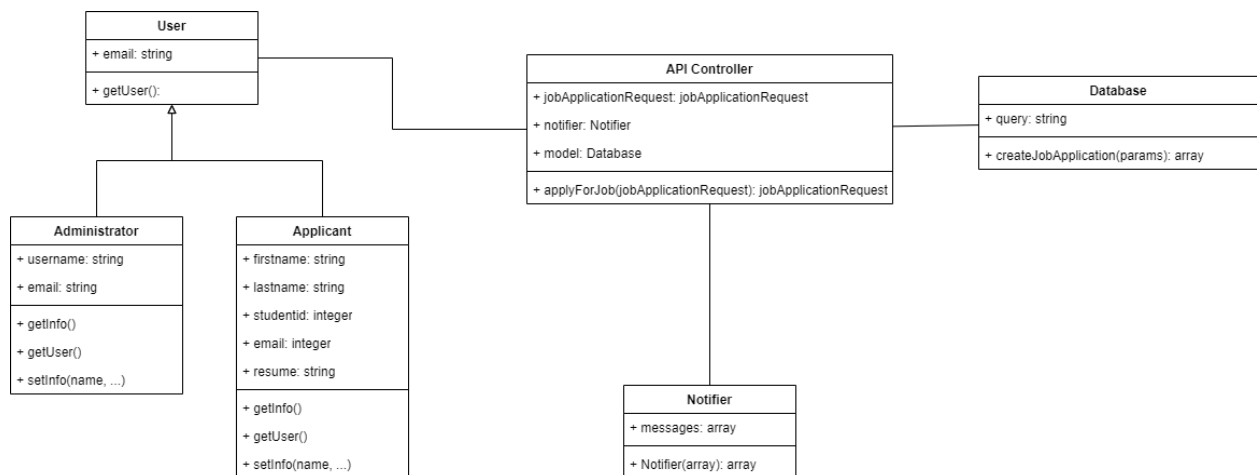
UC 5 - PostJob



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 phonenummer: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.

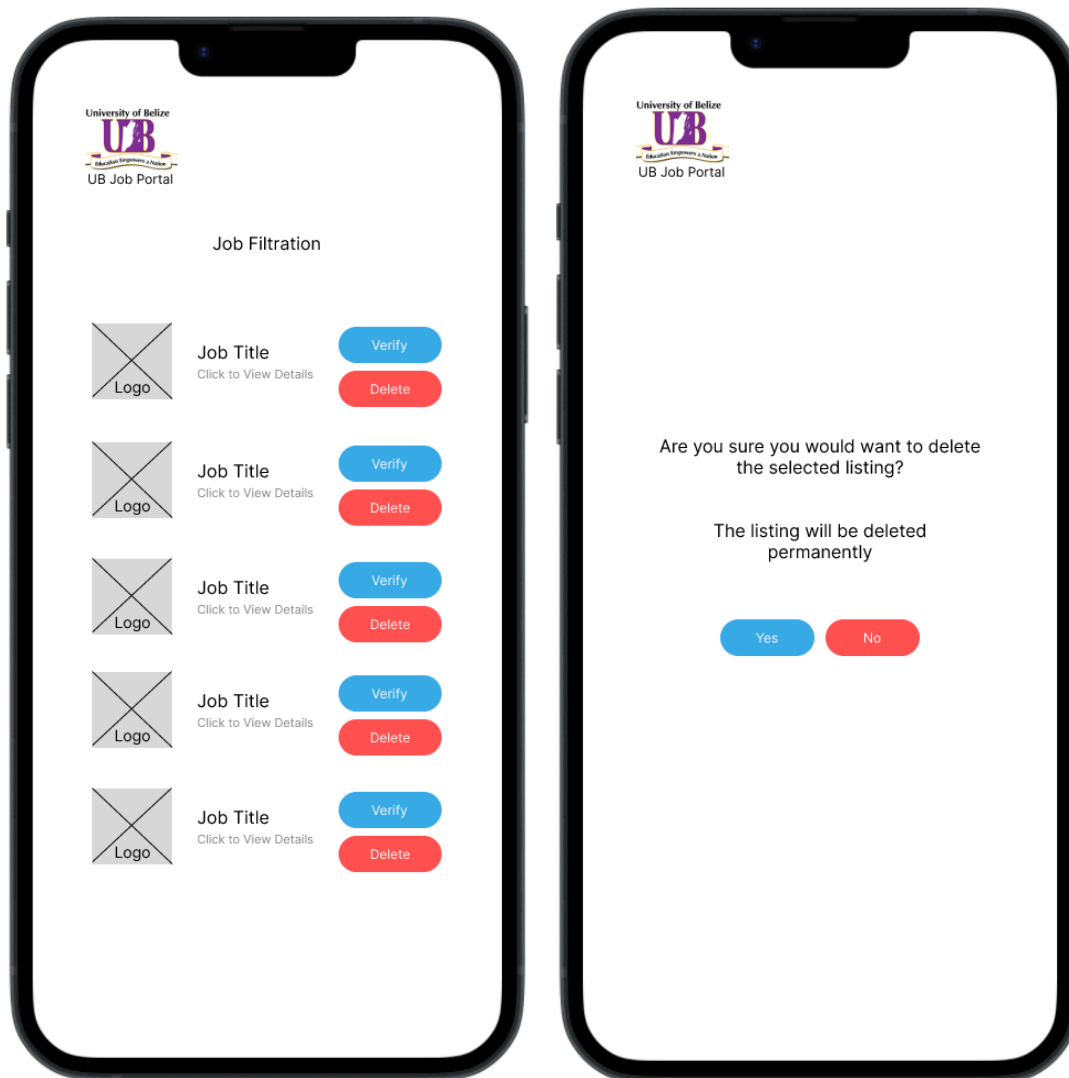
Synchronization: Synchronization mechanisms such as locks, semaphores, or mutexes are employed to ensure thread safety and prevent data races when accessing shared resources such as the database or message queues. Additionally, techniques like thread pooling may be used to manage and reuse threads efficiently.

Design Patterns:

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.



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 success/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

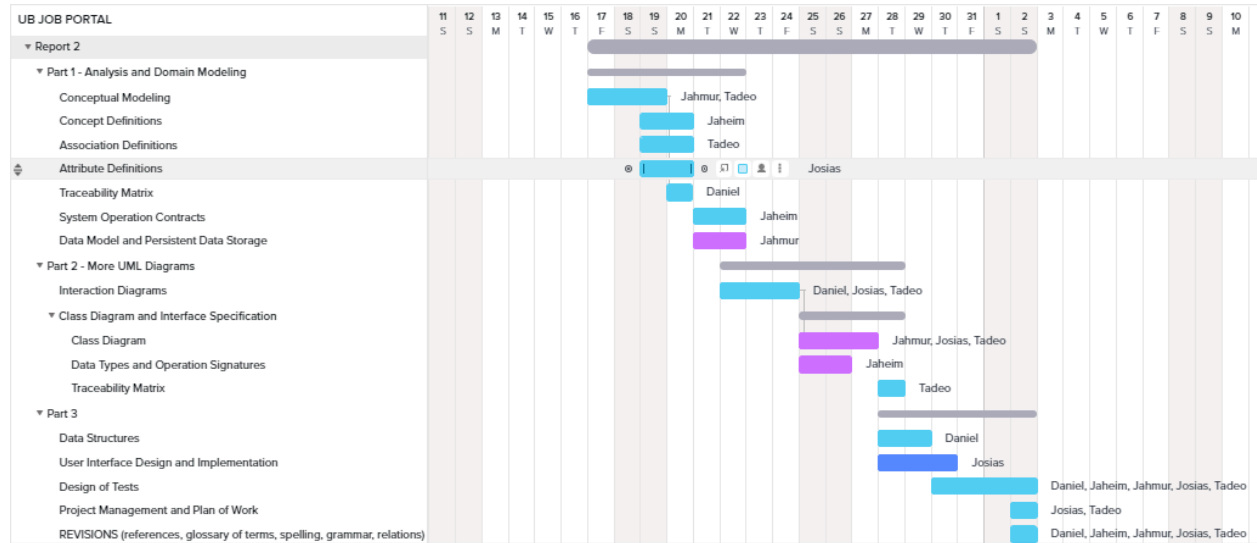
perform the requested action of inserting a user.	
---	--

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



Effort Estimation

Getting the Unadjust Actor Weight

Actor Classifications

Actor Type	Description of how to recognize the actor type	Weight
Simple	The actors is another system which interacts with our system through a defined application programming interface	1
Average	The actor is a person interacting through a text beast user interface or another system interacting through a protocol such as a network communication protocol	2
Complex	The actor is a person interacting via a graphical user interface	3

System Actors and their classifications

Actor	Description/Goal	Complexity	Weight
Applicant	Create an account Upload, delete, and update personal information and documents Apply for jobs Message employers about a job if considered	simple	1
Employer	Post job applications Upload, delete, and update company information Accept potential candidates Message candidates about a job and communicate further steps in the application process.	simple	1
Administrator	See system usage information	simple	1

$$UAW = 3x1 + 0x2 + 0x3 = 3$$

$$UAW = \#x\text{simpleweight} + \#x\text{averageweight} + \#x\text{complexweight}$$

Getting the Unadjusted Use Case Weight

Use Case Classifications

Use Case Category	Description of how to recognize the use case category	Weight
Simple	Simple user interface. up to one participating actor (plus initiating actor) number of steps for the success scenario: ≤ 3 if presently available, it's domain model includes ≤ 3 concepts	5
Average	Moderate interface design Two or more participating actors Number of steps for the success scenario: 4 to 7 if presently available, it's domain model includes between 5 and 10 concepts.	10
Complex	Complex user interface or processing Three or more participating actors Number of steps for the success scenario: ≥ 7 If available, it's the main model includes ≥ 10 concepts	15

System Use Cases

Name	Description	Category	Weight
UC 1 - Registration	Allows the system to create a profile of a user using their personal or business information.	simple	5
UC 2 - Authentication	Ensures that ONLY UB students and legitimate businesses are allowed to register and login.	simple	5
UC 3 - FilterJob	To allow the system to decipher which listings are legitimate.	average	10
UC 4 - ManageProfile	Allows system users to upload, edit, and delete their profile information.	simple	5
UC 5 - PostJob	To allow employers to post vacant job posts for applicants.	simple	5
UC 6 - SearchJobListing	To allow applicants to search through different job listings available.	simple	5
UC 7 - ApplyForJob	To allow applicants to apply for jobs by submitting their resume.	simple	5

UC 8 - Message	To allow both applicants and employers to communicate with each other via instant messaging.	average	10
UC 9 - ViewStudentListing	Allows employers to view applicants personal information that they made public, as well their resumes.	simple	5
UC 10 - AnalyzeStatistics	To allow the administrator to view, provide reporting & analytics of usage statistics.	average	10

$$UUCW = 7 \times 5 + 3 \times 10 + 0 \times 15 = \underline{65}$$

$$UUCW = \# \times \text{simpleweight} + \# \times \text{averageweight} + \# \times \text{complexweight}$$

Getting the TCF

TCF Table

Technical factor	Description	Weight	Perceived Complexity	Calculated Factor (WeightxPerceived Complexity)	Factors Used
T1	Distributed, Web-based system, because of ViewAccessHistory (UC-4)	2	3	$2 \times 3 = 6$	
T2	Users expect good performance but nothing exceptional	1	3	$1 \times 3 = 3$	
T3	End-user expects efficiency but there are no exceptional demands	1	3	$1 \times 3 = 3$	3
T4	Internal processing is relatively simple	1	1	$1 \times 1 = 1$	1
T5	No requirement for reusability	1	0	$1 \times 0 = 0$	
T6	Ease of install is moderately important (will probably be installed by technician)	0.5	3	$0.5 \times 3 = 1.5$	
T7	Ease of use is very important	0.5	5	$0.5 \times 5 = 2.5$	2.5
T8	No portability concerns beyond a desire to keep database vendor options open	2	2	$2 \times 2 = 4$	4
T9	Easy to change minimally required	1	1	$1 \times 1 = 1$	1
T10	Concurrent use is required (Section 5.3)	1	4	$1 \times 4 = 4$	4
T11	Security is a significant concern	1	5	$1 \times 5 = 5$	
T12	No direct access for third parties	1	0	$1 \times 0 = 0$	
T13	No unique training needs	1	0	$1 \times 0 = 0$	0

total = 15.5 technical factor sum

$$\begin{aligned}
 TCF &= 0.6 + (0.01 \times 15.5) \\
 &= 0.6 + 0.155 \\
 &= \underline{0.755}
 \end{aligned}$$

Getting the ECF

ECF Table

Environmental factor	Description	Weight	Perceived Impact	Calculated Factor (Weight x Perceived Impact)	Factors Used
E1	Beginner familiarity with the UML-based development	1.5	1	$1.5 \times 1 = 1.5$	1.5
E2	Some familiarity with application problem	0.5	2	$0.5 \times 2 = 1$	1.5
E3	Some knowledge of object-oriented approach	1	2	$1 \times 2 = 2$	2
E4	Beginner lead analyst	0.5	1	$0.5 \times 1 = 0.5$	
E5	Highly motivated, but some team members occasionally slacking	1	4	$1 \times 4 = 4$	4
E6	Stable requirements expected	2	5	$2 \times 5 = 5$	5
E7	No part-time staff will be involved	-1	0	$-1 \times 0 = 0$	0
E8	Programming language of average difficulty will be used	-1	3	$-1 \times 3 = -3$	-3

total = 11 environmental factor sum

$$\begin{aligned}\text{ECF} &= 1.4 + (-0.03 \times 11) \\ &= 1.4 + (-0.33) \\ &= \underline{1.07}\end{aligned}$$

Getting the UUCP

$$\begin{aligned}\text{UUCP} &= \text{UAW} + \text{UUCW} \\ &= 3 + 65 \\ &= \underline{68}\end{aligned}$$

Getting the UCPs

$$\begin{aligned}\text{UCP} &= \text{UUCP} \times \text{TCF} \times \text{ECF} \\ &= 68 \times 0.755 \times 1.07 \\ &= 54.9338 \\ &= \underline{55 \text{ UCP}}\end{aligned}$$

Effort Estimation

Productivity Factor (PF) = 28 hours per use case point

$$\begin{aligned}\text{Effort} &= \text{UCP} \times \text{PF} \\ &= 55 \times 28 \\ &= \underline{1,540 \text{ hours}}\end{aligned}$$

Reference

