COMP639 Studio Project – Semester 2 2025

Individual Assignment

Worth: 25%

Due: Sunday, 3rd August at 11.59pm.

Late Penalty: Work not received by due time attracts an immediate penalty of up to 25% of the marks available. No work will be accepted after **Tuesday, 5th August 2025 11:59pm**.

PROJECT OVERVIEW - INTERNSHIP MANAGEMENT SYSTEM - INTERNLINK

You have been tasked with developing a web-based internship management system for a fictitious start-up, **BrightPath Careers**, which is focused on reimagining early career recruitment.

BrightPath has identified a common issue: students often struggle to find relevant internships, while smaller companies lack a centralised platform to connect with emerging talent. To address this, you will build **Internlink**—a web application that streamlines the internship process for students, employers, and administrators.

You will use **Python with the Flask framework** to develop the application, with a focus on:

- Implementing a **secure, multi-user login system** with role-based access
- Designing intuitive interfaces for each user role:
  - **Students**: browse, apply for, and track internships
  - **Employers**: post internship listings, manage and track applications
  - **Administrators**: oversee users and platform activity

IMPORTANT

This is an **individual** assessment. You must not collaborate or confer with others. You may help others by verbally explaining concepts and making suggestions in general terms, but without directly showing or sharing your own code. You must develop the logical structure, the detail of your code and the database on your own, even if you are working alongside others. Code that is copied or shares a similar logic to others will receive zero marks for both parties.

The use of Artificial Intelligence (AI) tools, such as ChatGPT or GitHub Copilot, to complete this assessment is **prohibited**. You **must not** use AI to generate source code, database scripts, or any other deliverable submitted for marking. Assessment answers will be analysed for evidence of the use of AI and penalties may be administered.

The University policy on Academic Integrity can be found here.

SOFTWARE REQUIREMENTS

USER ROLES, SESSION HANDLING, AND ACCESS CONTROL

- **User Roles**:
  - Define three roles: **Student**, **Employer**, and **Admin** (administrator).
- **Session Handling and Access Control**:
  - Use Flask sessions to keep track of whether a user is logged in, and what their current role is.
  - Implement a system to limit access to certain pages or features based on the user's role.

WEB APP DESIGN

- **Responsive Design**:
  - The web app should adapt to different screen sizes. Most students will be using smartphones. Employers and Admins use a mix of smartphones, tablets, and laptop or desktop PCs.
  - Use **Bootstrap** to create a responsive UI.
- **Theming**:
  - Style your web app to match the theme.
  - Name your app "InternLink" and incorporate this name into your design. Make it clear to users that your web app is an intern management system.

HOME PAGE

- **Requirements**:
  - Include the name of the app ("InternLink").
  - Provide links for login and registration.
  - Include your name and student ID (the footer is a good place).

ROLE-SPECIFIC FUNCTIONS

Requirement	Student	Employer	Admin
Login and Registration			
Login: Create one login form for all users (i.e. students, employers, and admins should all log in using the same form, without having to specify their role). Store password hashes, not actual passwords.	✓	✓	✓
Logout	✓	✓	✓
Registration: Register new students to use the app. New users should always be registered as a student: people must not be able to register themselves as an employer or admin.	✓		

Student should provide username, email address, password, profile image (optional), full name, university, course, and resume upload in PDF format (optional). **Username and Password**: Ensure usernames are unique (i.e. no two accounts can have the same username). Passwords should be at least 8 characters long with a mix of character types. Make these constraints clear to users during registration. Do not store users' passwords directly in the database. Instead, hash passwords using the `flask_bcrypt` library and store those hashes. **Default Settings**: Set each new student role to "student" and their status to "active". **Note:** Employers and Admins will need to be manually created using SQL script.			
Internship Applications			
Browse internships: Filter internships by category, location, or duration.	✓		✓
Apply for internships: There should be an option to apply for a particular internship. The application page should have pre-filled information of the internship details (company name, title, description, location, duration and application deadline) and pre-filled information of the applicant details (full name, email address, university and course). It should also allow students to upload or replace resume (in PDF). There must be a text area for cover letter or additional note.	✓		
View status of internships applied for: Each application applied for should have a status of Pending, Accepted and Rejected. If the status is Accepted or Rejected, student can also view the reason for the decision.	✓		✓
Manage Internship Applications			
Browse Internships: Employers can only view internships posted by their organisation. **Note:** You do not need to implement features for Employers and Admins to create, edit, or delete internships. Instead, just provide SQL script to populate the database with a predefined list of internships.		✓	
View applications: View a list of student applicants for the internships that they have posted. They should be able to search application application's full name or by internship's title or by application status.		✓	✓
Manage application status: For each internship that employers have posted, they should be able to Accept or Reject applications with optional feedback.		✓	✓

Profile Management			
View and edit own profile details: Students should be able to view their full profile details and edit their full name, university and course. Students can also upload or replace resume (PDF only). For employers, they should be able to view their full profile details and edit full name, company name, company description, company website, company logo. Admins should be able to view their full profile and edit their full name and email.	✓	✓	✓
View, replace, and remove own profile image	✓	✓	✓
Change own password: Make sure the same password constraints you apply to new passwords during registration are also applied here, and that the new password is not the same as the current password.	✓	✓	✓
User Management			
View Users: View a list of all users with their role (student, employer, admin) and status (active/inactive). Allow searching or filtering by first name, last name, role and status.			✓
Change user status: (active/inactive)			✓

DATA REQUIREMENTS

- Database and Table Creation
  - Create your own script to set up the database and tables based on the ERD (Entity Relationship Diagram) in Figure 1.
- Database Population
  - Create your own script to populate the database with initial test data.
  - Store password hashes in the database, not actual passwords.
  - Include at least 20 students, 5 employers, and 2 administrators.
  - Add at least **20 internships** and **20 applications** in total. (not every internship needs an application, some could have multiple applications).

Note 1: When populating your database, please use realistic and meaningful data. Avoid using placeholder entries like 'user1', 'address1', or 'email1'. You can use a data generator such as https://generatedata.com/ to generate your test data.

Note 2: You may use generative AI to create internships and applications. **This is the only allowed use of generative AI in this assignment.** You may not use generative AI to create the database script itself: only to create realistic example issues and comments to include in your script.

- File Inclusion
    - You must include both the **database creation script** and the **database population script** in your GitHub repository **and** on PythonAnywhere.

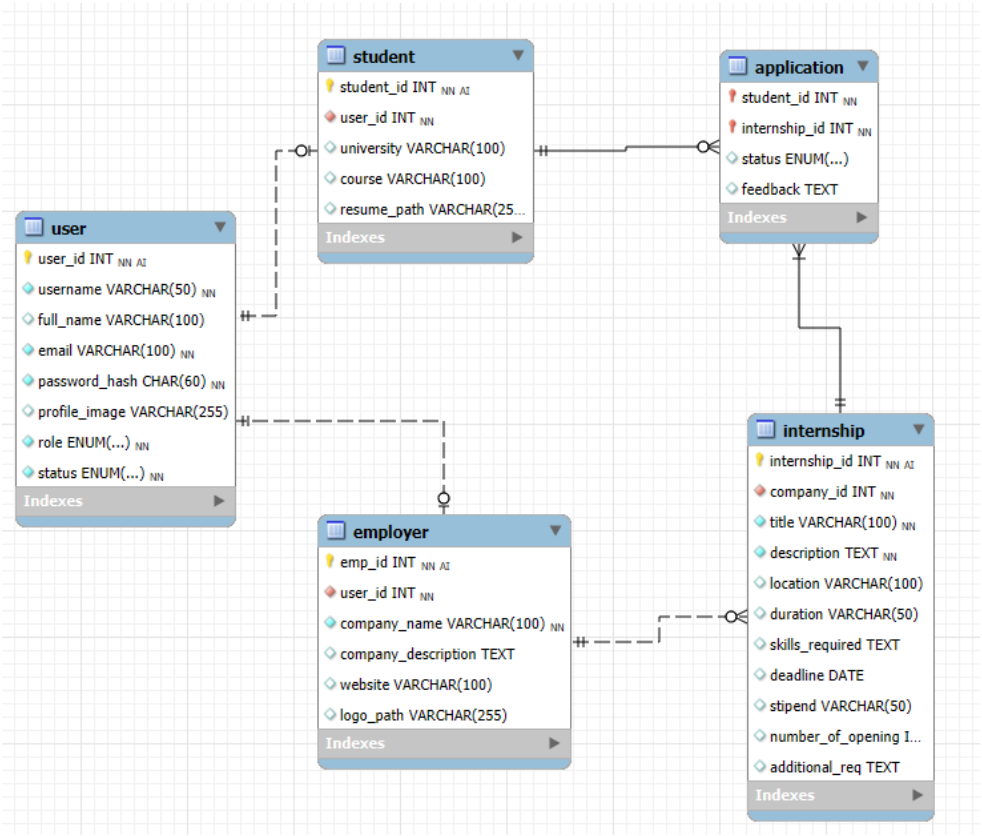

Figure 1: ERD illustrating the exact tables, fields, and relationships your database creation script must create. Note: You may add additional fields as needed.

Tip: To generate hashed passwords for your database population script, you may use the `password_hash_generator.py` script included with the Login Example project. You will need to install the `flask_bcrypt` library in your local development environment using `pip`. Make sure you follow all instructions given in the script, and ensure each user has a unique password.

The `password_hash` field in the `users` table (shown as CHAR(60) in Figure 1) is designed to store a `bcrypt` hash (60 bytes, which we store as characters). To be precise, you can set the data type for this field to `CHAR(60) CHARACTER SET utf8mb4 COLLATE utf8mb4_bin NOT NULL`.

Note: You can store user's profile image as a static file, and the `profile_image` column of the `users` table will contain the filename. Images themselves are not stored in the database. We chose this approach because it simplifies the deployment of your app to PythonAnywhere.

PROJECT REQUIREMENTS

1. TECHNOLOGY STACK

- Build your web app using Python, Flask, Bootstrap CSS, JavaScript, and MySQL.
- You may use CSS to override the default Bootstrap colours, but **do not** write full custom CSS layouts.
- **Do not** use SQLAlchemy, ReactJS, or other technologies.

2. GITHUB REPOSITORY

- Create a new private repository named "`InternLink`"
    - Add a README file with detailed instructions on how to deploy and use your web app, including any setup or configuration steps required. **Do not** copy/repeat the sample README.md provided with the login system example — instead, adapt the content to accurately reflect your own application, and remove any irrelevant information.
    - Ensure your repository has a `.gitignore` file to exclude the virtual environment and your `connect.py` file, which stores your database connection details (different on your local machine and PythonAnywhere).
- Include the following in your repository:
    - All Python, HTML, image files, and any other necessary files for the web app.
    - A `requirements.txt` file listing all required pip packages.
    - Two MySQL scripts: one for database creation and one for record population.
    - Add GitHub user "`lincolnmac`" (`computing@lincoln.ac.nz`) as a collaborator to the repository.

3. HOSTING ON PYTHONANYWHERE

- Host your system, including the database, on PythonAnywhere.
- Add "`lincolnmac`" as your "`teacher`" via the site configuration.

4. SUBMISSION

- Submit the COMP639 Web App Hand-In Sheet via the link on the COMP639 Akoraka | Learn page.
- Include the following in your submission:
    - Your PythonAnywhere URL.
    - Your GitHub repository URL.
    - Usernames and passwords for different user roles for testing purposes.
    - Confirmation that certain files have been saved in your GitHub repository.

Note: The Hand-In Sheet must be submitted as an Excel file. Submissions in PDF or any other format will not be accepted. Additionally, please ensure that all usernames and passwords

are correct. Please bear in mind that, if we are unable to log in using the credentials you supply, we will not be able to assess your assignment and may delay the marking process.

IMPORTANT: Do NOT make any changes to your app on GitHub or PythonAnywhere after the submission date until your marks have been released. Any changes will be considered the same as a late submission and may be penalised with a late penalty of up to 100% of your mark.

MARKING CRITERIA

The functionality and design of your web app will only be assessed on PythonAnywhere. If your app does not run on PythonAnywhere, it will not receive a mark in those areas.

Your code, database scripts, and `README` file will be viewed on GitHub. Make sure you have shared your repository with the "`lincolnmac`" account.

Functionality (50)				
Registration, Login and Logout (10)				
Registration	Registration form not implemented or broken. (0)	New visitors can register an account. (2)	All required details included (username, email, password, full name, university and course). Password constraints enforced. New accounts are defaulted to students. (3.5)	Intuitive registration process with useful validation hints and error messages. Double-entry of password for confirmation. No notable bugs or usability issues. (5)
Login and Logout	Existing users cannot log in. (0)	Existing users can log in. (2)	Existing users (including newly registered users) can log in and log out. All users log in the same way, regardless of role. (3.5)	Intuitive login and logout functionality with no unnecessary steps. No notable bugs or usability issues. (5)
Internship Applications (10)				
Browsing Internship	Users cannot see the internship list. (0)	Users can see the internships, but filters are missing or broken. (1.5)	Browsing and partial filtering works. May be missing category/location/duration filters. (3)	Full filtering and smooth browsing interface. Students see all internships. No notable bugs or usability issues. (4)

Internship Application	Application feature not available. (0)	Students can apply, but form lacks key elements (e.g., pre-filled data, resume). (1)	Form is pre-filled. Resume upload and cover letter work with minor issues. (2)	Pre-filled internship and applicant data. Resume upload/replace in PDF. Cover letter supported. No notable bugs or usability issues. (3)
Application Status Tracking	Students cannot see application statuses. (0)	Students see only their applications, not status. (1)	Statuses (Pending, Accepted, Rejected) shown but feedback not provided. (2)	Students can see application statuses clearly along with feedback. No notable bugs or usability issues. (3)
Manage Internships (10)				
View Internships	Employers cannot see any internship postings. (0)	Employers can see internships but are shown all internships, not limited to their organization. (1)	Employers can view internships posted only by their organization, but the interface may have usability issues or bugs. (2)	Employers can view internships posted only by their own organization with a smooth, bug-free interface. (3)
View Internship Applications	Employer cannot view any applications. (0)	Employer can view applications but cannot search or filter by applicant full name, internship title, or status. (1.5)	Employer can view applications and perform basic search on at least one criterion (e.g., applicant full name), but other filters/search options are missing or buggy. (3)	Employer can view applications with fully functional search/filter by applicant full name, internship title, and application status. Interface is intuitive and bug-free. (4)
Manage Application Status	Employer cannot change application status. (0)	Employer can Accept or Reject applications, but cannot provide feedback, or status changes are unreliable. (1)	Employer can change status and provide feedback, but feedback is optional or limited, or interface has minor issues.(2)	Employer can reliably Accept or Reject applications with optional detailed feedback. Status updates reflect immediately, and interface is smooth with clear confirmation. Interface is intuitive and bug-free. (3)
Manage Profiles (10)				
View and Update Profile Details	Not recognisably implemented, or does not function as specified. (0)	Users can update their email address, full name, and other fields depending on the user role. (2)	Users can update all details. Previous details are clearly visible during the update. (3.5)	Users can update all details. Previous details are directly editable. No notable bugs or usability issues. (5)

View and Update Profile Image	Not recognisably implemented, or does not function as specified. (0)	Users can add a profile image, and see that image on their profile. (1)	Users can add, replace, or remove their profile image. New users begin with a default image or placeholder. (2)	Users can add, replace, or remove their profile image. Images are displayed correctly regardless of size and shape. No notable bugs or usability issues. (3)
Change Password	Not recognisably implemented, or does not function as specified. (0)	Users can change their password. (1)	Users can change their password. The same constraints applied during registration are applied here. Password is not displayed on screen. No notable bugs or usability issues. (2)	
User Management (10)				
Search and View User Profiles	Not recognisably implemented, or does not function as specified. (0)	Admins can view a list of users and view the profile of any user. (2)	Admins can view a list of users, searchable by username, and view the profile of any user. (4)	Admins can view a list of users, searchable by username, full name, and view any user's profile. No notable bugs or usability issues. (6)
Change User Status	Not recognisably implemented, or does not function as specified. (0)	Admins can change the status of any user (student, employer, or admin) from "active" to "inactive" or vice-versa. (2)	Admins can change the status of any user via an appropriate control, which defaults to a user's current status. No notable bugs or usability issues. (4)	
Design (15)				
Format and Layout	No visible formatting or layout (plain HTML). (0)	Some appropriate formatting and layout. (1.5)	Appropriate formatting and layout, consistent across most pages. (3)	Simple and attractive layout, consistent across all pages of the app. (4)
Colour Scheme and Style	No consistent colour scheme or style. (0)	Generally consistent across most pages. (1)	Consistent across most pages and matches the theme of the app. (2)	Consistent across all pages and matches the theme of the app. (3)
Responsive Design	Design does not respond to changes in screen size. (0)	Some design elements are responsive to changes in window size. (1)	Most design elements adapt well to changes in window size on desktop. (2)	Most design elements adapt well to changes in window size, and adapt well to common smartphone/tablet screen sizes. (3)
Role-Based Access Control	Users can access functions for other roles. (0)	Users can only access functions appropriate to their role. (2)	Users can only see (in menus, etc) and access functions appropriate to their role. (3.5)	All users share the same UI where appropriate, but can only see and access functions appropriate to their role. (5)

Implementation (35)				
Source Code (20)				
Structure and Organisation	All code in one file, with no clear organisation. (0)	Code is organised into multiple files, with some logical structure. (4)	Code is well organised into multiple files, uses functions and constants to avoid unnecessary repetition, and has some consistency in naming conventions. (7)	Code is well organised, avoids unnecessary repetition, uses good naming conventions for variables and functions, and follows other best practices. (10)
Security	No password hashing, session management, or role-based access control. (0)	Passwords are hashed. Some form of session management and role-based access control. (2)	Passwords hashed. Good, generally consistent approach to session management and role-based access control. (3.5)	Passwords hashed. Good, highly consistent, and modular approach to session management and role-based access control, using functions and/or custom decorators to reduce repetition. (5)
Error Handling and Validation	Not recognisably implemented. (0)	Some error handling and/or validation. (2)	Appropriate error handling and some server-side validation. (3.5)	Appropriate error handling and consistent server-side validation. (5)
Documentation (10)				
Commenting	No useful comments included in source code. (0)	Some useful comments included in source code. (3)	Consistent and useful comments on modules, functions, and any sections of code that need explanation. (5.5)	Consistent and useful comments, with Python Docstrings for modules and functions. (8)
GitHub README	Not included or a copy/repeat of the Example Code ReadMe. (0)	Explains how to set up and use the project. (1)	Provides a detailed guide to setting up and using the project. (2)	
Database (5)				
Table Structure	No table creation script provided. (0)	Implemented based on the ERD, but with one or more incorrect tables. (1)	Implemented based on the ERD, with all required tables. (1)	Implemented based on the ERD, with all required tables, fields, and relationships. (2)

SQL Script for Record Creation	Not provided. (0)	Script included to create students, employers, administrators, and some issues/comments. (1.5)	Script correctly creates at least 20 students, 5 employers, 2 administrators, 20 internships, and 20 applications. Test data is relevant and realistic. (3)