# IFN582 Assessment 3 - Fully Developed Web Application: Details

## Assessment 3: Fully Functional Image Sharing and Purchase Web Application

NB: This is a **group assessment**, and rules apply regarding group formation, deadlines, and peer evaluation. Canvas will not allow any individual to submit the assignment. Only students who are part of a group can submit the assignment.

Please refer to the **Group Formation and Rules page (https://canvas.qut.edu.au/courses/22634/pages/group-formation-and-rules)** for complete details and important deadlines.
*All students are expected to read and follow these guidelines carefully.*

## Overview

### Task Description:

This assessment task requires you to build a **Fully Functional Image Sharing and Purchase Web Application (virtual gallery)** based on one of the **initial designs (Assessment 1)** and **the corresponding database design (Assessment 2)**. The application can be developed for one of the following domains:

1. **Photography Studio**: Photographers display their portfolios and allow customers to book sessions.
2. **Art Lease**: Artists and galleries display artworks that can be rented.
3. **Stock Image Purchase**: Stock images can be uploaded, displayed, purchased, and/or downloaded.

The goal of this task is to demonstrate your ability to collaboratively develop a **fully functional web application** that allows users to browse available items, view details, manage orders/bookings, and complete the purchase or booking process. This **group assessment** is designed to test your ability to implement a complete web application with front-end and back-end integration, database integration, and access control while ensuring usability and responsiveness.

**This assessment is due on October 24, 2025, at 11:59 pm. This is an assignment for the purposes of an official extension.**

### Unit Learning Outcomes (ULOs) Assessed:

**ULO2:** Apply standard querying techniques to retrieve and manipulate data.

**ULO3:** Use existing frameworks to develop and implement web applications with a rigorous database design to meet given requirements, including being ethical and culturally sensitive.

**ULO4:** Apply visual and written communication techniques to explain how the proposed design and solution meet user and data management needs.

**ULO5:** Collaborate and work effectively in a team environment, apply teamwork concepts critically reflecting upon personal performance, and the performance of others.

| Estimated time for completion | Weighting | Group or Individual | How I will be assessed |
|---|---|---|---|
| 30 hours | 50% of Final Grade | Group | Using a 7-Point Grading Scale |

**Note:**

**The use of generative artificial intelligence (GenAI) tools is not permitted** in this assessment task, as you must demonstrate your own skills and knowledge. Ensure that all submitted work is your own and maintains academic integrity standards.

You may be asked to demonstrate authorship of your assessment. **Find out more about keeping good records to authenticate authorship (https://qutvirtual4.qut.edu.au/documents/d/student/keeping-good-records-to-authenticate-authorship) .** The Academic Integrity Policy (https://mopp.qut.edu.au/document/view.php?id=146) describes penalties related to the unauthorised use of GenAI.

## What you need to do

### Assignment Objectives:

By completing this assessment, you will:

- Apply **Model-View-Controller (MVC)** principles for organizing application logic and data models.
- Demonstrate **CRUD (Create, Read, Update, Delete)** functionality across **core features**, such as items**,** galleries, bookings, and checkout.
- Ensure **database integration**, **form validation**, and **error handling** across all pages.
- Provide a **responsive** and **professional** user interface.
- Implement **authentication and access control** for users, vendors, and admins.
- Deliver a **group demonstration** showcasing the application's features and answering technical questions.

## Requirements for Your Solution:

Your application must implement the following **pages and features**:

- **Home Page** (Search, Filter, Browse):
  - Display images/artworks/services dynamically fetched from the **database**.
  - Support **search and filtering** by category, vendor/photographer, or keyword.
  - Ensure responsive design across devices.
  - Include a functional **navbar/header and footer**.
- **Vendor Gallery & Management**
  - Public-facing gallery for each vendor (photographer, artist, or stock contributor).
  - Vendor-only management page to **upload, edit, or delete** images/artworks.
  - Access control to ensure only vendors can manage their own gallery.
- **Item Details Page**:
  - Show detailed information: image, title, description, price, availability.
  - Options to:
    - Add item(s) to the basket.
    - Book a session (for Photography Studio).
    - Select rental duration & delivery location (for Art Lease).
  - Use **form validation** and provide feedback for invalid inputs.
- **Checkout Page**:
  - Display selected items with options to **add, update, or remove items**.
  - Handle **empty basket** scenarios by preventing checkout if no items are selected, showing a clear message, and providing a **"clear basket" button**.
  - Calculate and display **totals dynamically**.
  - Collect user details, including **name, contact, and payment information**.
  - Allow selection of **delivery or booking methods** (e.g., photography session booking, art lease with rental/delivery, stock image purchase).
  - Show a complete **order summary with accurate totals** before final submission.
  - Validate all form inputs and handle invalid data gracefully with clear feedback messages.
  - Ensure the layout is **responsive, intuitive, and professionally styled**.
- **Authentication and Access Controls**:
  - Implement a complete user authentication system, including **Registration, Login,** and **Logout**, using hashed passwords and session-based authentication.
  - Restrict admin access for managing all content, vendors, and orders.
  - Restrict vendor access for managing personal gallery.
  - Restrict customer access for browsing, booking, and purchasing.
  - Implement access control using **custom decorators** (e.g., `@admin_required`) or **Flask-Login session-based role checks**.
  - Do not use **Flask-Admin**.
- **Error Handling:**
  - Handle and display custom error pages for at least:
    - 404 Not Found.
    - 500 Internal Server Error.

  - Use an error.html template.
  - Flask `@app.errorhandler` should manage redirection and display.
- **Professional User Interface:**
  - Your interface must:
    - Be responsive across screen sizes.
    - Have a consistent color scheme, fonts, and layout.
    - Avoid overlapping or broken elements.
    - On all pages, ensure that you include:
      - Navigation bar/header.
      - Page footer.
      - Error-free forms and buttons.
      - Clear spacing and alignment.
- **Database Integration and Sample Data:**
  - Your application must be built on the **data model developed in Assessment 2**, with any necessary refinements to support full functionality.
  - The database must include:
    - **At least 15 individual items in total** (e.g., photographs, artworks, or stock images), distributed across a minimum of **2 different item categories.** For example, "Wedding Photography" and "Event Photography" for a studio, "Paintings" and "Sculptures" for art lease, or "Nature" and "Architecture" for stock images would be two distinct categories.
    - **At least 6 users,** including:
      - **2 admin users** with full system privileges.
      - **2 vendor users** (e.g., photographers, artists, or stock contributors) responsible for managing their own gallery.
      - **2 customer users** who can browse, book, lease, or purchase items.
    - **At least 3 completed transactions** (orders, bookings, or leases), each demonstrating a **different purchase or booking method.** Examples:
      - A photography session booking.
      - An art lease with a rental period and delivery address.
      - A stock image purchase and download.
- **In-Class Demonstration:**
  - Each group will present its web application during the tutorial session in Week 13.
    - Each group will have **4 minutes** for the live demonstration and **8 minutes** for questions and answers.
    - Any group member may lead the demo, but the tutor will ask questions to **each group member**.
    - All group members are expected to have a good understanding of the full application, including back-end and front-end features.
  - **Note:** There is **no requirement to prepare PowerPoint slides** for the demonstration. The purpose of the in-class demo is to walk through the working web application, show its features, and discuss the design and technical choices made.
- **Peer Evaluation (via FeedbackFruits)**:
  - Each group member must:

- Submit **individual peer reviews** via FeedbackFruits.
- Provide a **self-reflection on their own contributions** as well as reflect on those of team members.
- Assess teamwork, participation, and communication.

### Permitted Tools and Resources:

- **Frameworks:** You must use only the frameworks and libraries taught and discussed in tutorials or explicitly permitted, such as **Flask**, **Jinja2**, **Flask-WTF**, **WTForms**, **Flask-MySQLdb**, **MySQLClient**, **bootstrap-flask,** and Python standard libraries (e.g., `datetime`, `os`, etc). The complete list of permitted packages is provided in **requirements-1.txt (https://canvas.qut.edu.au/courses/22634/files/6634539?wrap=1) (https://canvas.qut.edu.au/courses/22634/files/6634539/download?download_frd=1)** on Canvas. Use of high-level frameworks, automation tools, or alternative libraries, including **Flask-Admin**, **Django**, or similar, is strictly prohibited.
- **Front-End:** Use only HTML and CSS, including Bootstrap 5.3.

### Important Notes:

- **Original Design**: Reusing templates or external solutions such as Wix, Squarespace, or Bootswatch is strictly prohibited. However, using code snippets from resources like **w3schools (https://www.w3schools.com/)** is allowed.
- **Original Work:** Collusion with other students and the use of Generative AI tools are forbidden.
- **Responsiveness**: Ensure your design works well on desktop, tablet, and mobile screens. Navigation between pages should also be simple and intuitive. Bootstrap components should be used.
- **Dependencies:** Markers will evaluate your work by observing the HTML pages in Google Chrome. Your Python Flask application will be run locally, so please ensure that you have taken care of all the necessary dependencies (there should be no unexpected imports or Python libraries used besides those prescribed above).
- **Debug Mode:** Ensure that `app.debug = False` in your final submission. The application must simulate a production-ready deployment.

## What to submit

Submit the following deliverables in **Canvas**:

- **Web Application Code (ZIP File):**
  - Organised project folder containing all **HTML, CSS, Python scripts**, and other assets.
  - Include a **MySQL database** pre-populated with sample data.
  - Exclude any **virtual environments** and **external dependencies** from submission.
- **Declaration Form (PDF):**
  - Include:

    - Full names and student IDs of all group members.
    - A description of each member's role and contribution to the project.
  - **Note:** Please use the following template and submit your declaration form: **Assessment 3 Declaration Form.docx (https://canvas.qut.edu.au/courses/22634/files/6802008?wrap=1) (https://canvas.qut.edu.au/courses/22634/files/6802008/download?download_frd=1)**
- **Peer Evaluation:**
  - Submit your peer evaluation via FeedbackFruits using the following Canvas link: **Group Formation and Rules (https://canvas.qut.edu.au/courses/22634/pages/group-formation-and-rules)**
  - **(https://canvas.qut.edu.au/courses/22634/assignments/201433)** Follow the instructions provided on the linked Canvas page.

## How to submit

- Ensure that you submit your declaration form and your zip file as **separate files** via Canvas – the submission link will allow you to upload multiple files.
- For your web application, submit a **single zip file** named **"**[groupID]_582_A3.zip" via Canvas.
- For your declaration form, submit a single file named **"**[groupID]_A3.pdf" via Canvas.
- Only **one group member** should submit the above files on behalf of the group, ensuring that **all group members names and IDs** are clearly mentioned in the PDF.
- Ensure all required files are included in the correct folder structure (rename or add files as necessary):

```
[groupID]_582_A3.zip
├── run.py
├── README.txt          (brief setup instructions)
└── project/
    ├── static/
    │   ├── css/
    │   └── img/
    ├── templates/
    ├── forms.py
    ├── models.py
    ├── views.py
    ├── __init__.py
    └── database.sql     (pre-populated)
```

- Double-check that all files required for running the application are included.
- Document how to run the project in the README.txt.

## Important

After uploading your assignment, you should click the **Submit** button. You will see a submitted date and time appear at the top of your screen. Unless you see this date and time, **your assignment has not been successfully** submitted and you should contact HiQ at askqut@qut.edu.au (mailto:askqut@qut.edu.au) or 3138 2000. You can also check that your assignment has been submitted by clicking on the **Grades** link. You should see the **date** and **time** (in the "Submitted" column) beside the **name** of the assessment task.

## Feedback

Under normal circumstances, you will receive marks for each criterion via a Canvas rubric within 10-15 working days of submission. Click on Grades to see your results. Usually the reason for each choice of mark is self-evident, the marker will include some written feedback about your performance. You should use this feedback to strengthen your performance in the next assessment item.

## Moderation

All staff who are assessing your work meet to discuss and compare their judgments before marks or grades are finalised.