**Course Work Answer Book**

**UNIVERSITY COURSE WORK**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| REGISTRATION NUMBER | | | | | | | | | VU-BCS-2403-0834-DAY | | | | | | |
| Title of The Program (eg BBA, BSC, BPH, BSWA) | | | | | | | | | | | | | BCS | | |
| Bachelor of Science in Computer Science | | | | | | | | | | | | | | | |
| Department | | | | Other Depts in Faculty of Science and Technology | | | | | | | | | | | |
| Faculty | Faculty of Science and Technology | | | | | | | | | | | | | | |
| Year Of study (YrI , YrII, YrIII, or YrIV) | | | | | | | | | | | 2 | | | | |
| Module Code and Name | | | | | | | 2103 ST | | | | | | | | |
| Systems Analysis and Design | | | | | | | | | | | | | | | |
| Semester | | | 1 | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | |
| Retake: | | Yes | | |  | | | No | |  | | (Tick whichever is applicable) | | | |
| Date of Course Work | | | | | | Mon May 05 2025 00:00:00 GMT+0300 (East Africa Time) | | | | | | | | | |
|  | | | | | |  | | | | | | | | | |
| **DIRECTIONS TO CANDIDATES (Turn to page ii for more instructions).** | | | | | | | | | | | | | **FOR USE BY EXAMINERS ONLY** | | |
| **Question Number** | **Internal Examiner** | **External Examiner** |
| 1. Leave margin blank. 2. Begin each answer on a fresh page. 3. Write the number of each question and theCandidate's Number at the top of each page. 4. Write the numbers of the questionswhich you have attempted, with subsections where necessary, in the spacesprovided below | | | | | | | | | | | | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **TOTAL** |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **NUMBER OF QUESTIONS** you have answered in the order in which you have written them | | | | | | | | |
|  |  |  |  |  |  |  |  |  |

**How and where should I submit my Course Work script?**

Every student will be required to submit their Course Work via [VClass Students Portal](https://vclass.ac/) E.g. you go to [www.vclass.ac](http://www.vclass.ac) and login, to your account, then on the left sidebar menu **click on Course Work**.

Under Course Work you will see the following: -

1. Instructions for that particular Course Work with time required to finish your Course Work as per instructions,
2. A student will be required to download the question paper and the answer sheet provided by the university within the same module Course Work, or a student can be required to attempt structured questions within the system depending on how the Course Work was set.
3. Submission of answered questions is done,
4. Student is required to click to **consent** to show that the answered Course Work belongs to them.
5. **Note** that if Course Work is for download, a student will be required to download the question paper and answer sheet, do their Course Work within the given stipulated time.
6. Required to scan and upload back the answered booklet through the same portal as per format available.
7. Course Work uploaded will directly be received by the Registry department.
8. Students here are required to use [VClass e-Learning system](https://vclass.ac)for all Course Work and for any failure they can contact the Registry department for guidance.
9. No late submission will be accepted.

**Avoid any malpractice because this will attract severe penalties such as invalidating the answered script whose consequences will attract retakes.**

**VICTORIA UNIVERSITY EXHIBITION TICKET & ATTENDANCE MANAGEMENT SYSTEM PROPORSAL.SSS**

**TEAM MEMBERS**

AKULLO VANESSA

BELINDA MATTHEWS

OCEN SEDRICK

**21ST May,2025.**

**1. Executive Summary**

This proposal outlines the development of a comprehensive Ticketing and Attendance Management System for the Victoria University Faculty of Science and Technology Exhibitions. Designed to replace the current manual, error-prone processes, this system will automate guest registration, manage event and session details, oversee seating arrangements across varied categories and pricing, facilitate booking specific seats for specific sessions, and enable attendance tracking. The system will provide a centralized, accurate database (developed in MySQL) and a user-friendly interface (structured using HTML/CSS via VSCode) to eliminate issues like misplaced badges and improve efficiency, real-time visibility, and communication capabilities for event organizers. The development process utilized Lucidchart for initial design visualization.

**2. Introduction and Problem Statement**

The current manual handling of exhibition logistics at Victoria University, particularly guest registration, seating management, and attendance tracking, poses significant challenges. The incident involving Eng. Mutale Peter's misplaced VIP badge serves as a clear example of the chaos and unreliability inherent in paper-based lists and the lack of real-time information. Managing 602 seats with four distinct categories, unique numbering patterns, and varied pricing, along with tracking individual seat bookings across multiple sessions within a single event, is complex and highly susceptible to human error. Furthermore, the manual system makes efficient follow-up communication with registered guests difficult. A dedicated, automated system is essential to address these inefficiencies, ensure accuracy, and streamline the entire exhibition management process. The initial steps of creating a web interface using VSCode and designing the database in MySQL aim to provide the foundation for this solution.

**3. System Objectives**

The core objectives of the proposed system are to:

⦁ Establish a secure, centralized database (in MySQL) for all relevant exhibition data (guests, events, sessions, seats, bookings).

⦁ Automate guest registration through a web interface, including the assignment of unique identifiers and capturing comprehensive contact and affiliation details.

⦁ Accurately define, schedule, and manage exhibition events and their associated sessions, accommodating multiple sessions per event, including morning and afternoon slots.

⦁ Store and manage the detailed seating plan in the database, including 602 seats, their four categories, unique numbering patterns, and associated pricing.

⦁ Enable guests or administrators to book specific, available seats for specific sessions via the interface.

⦁ Provide real-time tracking of seat availability for each session.

⦁ Support the recording and tracking of guest attendance for each session they are booked for.

⦁ Generate necessary reports from the MySQL database on guest registration, bookings, seat availability, and attendance.

⦁ Significantly reduce manual data entry errors and improve the reliability of information.

⦁ Provide a foundation for efficient communication and follow-up with registered guests.

**4. Proposed System Requirements**

**4.1. Functional Requirements:**

⦁ Guest Management:

⦁ The system shall allow the capture and storage of guest details: Full Name, Affiliation (organization/institution), Email Address, Phone Number, and Title via a web form.

⦁ The system shall automatically generate and assign a unique GuestID to each new guest record upon submission to the database.

⦁ The system shall support querying and displaying guest information from the database for administrative purposes and follow-up communication.

⦁ Event Management:

⦁ The system shall allow administrators to define and manage exhibition events in the database, including capturing the Event Title and Event Type (e.g., Project Showcase, Research Poster, Keynote Session).

⦁ Session Management:

⦁ The system shall allow administrators to define and manage individual sessions in the database, capturing Session Type (e.g., Workshop, Presentation), Time Slot, and Day Part (Morning/Afternoon).

⦁ The system shall link each session to its parent event using a foreign key relationship in the database.

⦁ The system must support multiple sessions being associated with a single event.

⦁ Seating Management:

⦁ The system shall store details for all 602 seats in the MySQL database.

⦁ The system shall categorize seats into the four specified categories within the database structure.

⦁ The system shall store the unique seat number patterns and the associated pricing for each category in the database.

⦁ The system must accurately track the booking status (availability) of each specific seat for each specific session using the Bookings table data.

⦁ Booking Management:

⦁ The system shall allow a guest or administrator, through the interface, to select an event, then a session, and then choose from the available seats for that session (by querying the database).

⦁ Upon selection and confirmation via the interface, the system shall create a booking record in the database associating a specific guest (GuestID) with a specific seat (SeatID) for a specific session (SessionID).

⦁ The system shall record the date and time the booking was made in the database.

⦁ The system shall prevent the same seat from being booked by another guest for the same session by checking for existing bookings in the database.

⦁ The system shall store and allow administrators to update the status of a booking (e.g., Confirmed, Attended, Cancelled) in the database.

⦁ Attendance Management:

⦁ The system shall allow administrators, likely via an administrative interface, to record or mark the attendance of guests for the sessions they are booked for, updating the booking status in the database.

⦁ Reporting:

⦁ The system shall generate reports by querying the MySQL database, such as a list of all registered guests, bookings for a specific event or session, a view of seat availability for a selected session, and attendance reports for sessions.

⦁ User Interface:

⦁ The system shall provide a user-friendly web interface (developed using HTML/CSS via VSCode) for guest registration and booking interaction.

⦁ The system shall provide an administrative interface (design to be determined, could be web-based or desktop) for managing events, sessions, seats, guests, bookings, and attendance, interacting directly with the MySQL database.

**4.2. Non-Functional Requirements:**

⦁ Data Integrity: Ensure all data stored in the MySQL database, especially guest information, booking linkages, and seat availability status, is accurate and consistent through database constraints and application logic.

⦁ Reliability: The system must function reliably, particularly during periods of high activity like guest registration and attendance check-in. The MySQL database should be configured for stability.

⦁ Usability: The interface for both guests and administrators must be intuitive and easy to use, requiring minimal training. The HTML structure provides a starting point for this.

⦁ Performance: The system should respond quickly to user requests, such as checking seat availability and creating bookings, even with the full dataset of 602 seats and numerous bookings, which relies on efficient MySQL queries.

⦁ Security: Appropriate measures must be in place to protect guest data stored in MySQL and prevent unauthorized access to administrative functions and the database.

⦁ Maintainability: The system's code (developed in VSCode) and database structure (designed using Lucidchart and implemented in MySQL) should be well-organized and documented to allow for future updates and modifications.

**6.Pre-Implementation Review**

Before commencing the coding phase, significant time was spent in planning and design. The key activities included analyzing the case study requirements, identifying the main entities and their relationships, and designing the database schema using Lucidchart. A crucial understanding gained during this phase was the complex relationship between Guests, Seats, and Sessions, which necessitated the creation of a dedicated Bookings linking table in the MySQL database. Initial thoughts on data import highlighted the need to convert the provided Excel file containing seat information into a compatible format like CSV before attempting to load it into MySQL, identifying a potential point of failure if not handled correctly. Defining the primary and foreign keys in Lucidchart before writing SQL code for MySQL was instrumental in visualizing the structure and ensuring data integrity constraints could be applied. The plan involved creating the database and tables in MySQL first, then building the basic HTML forms in VSCode for data input, with the understanding that a backend component would be needed to connect these layers.

**7. Post-Implementation Review**

**⦁ What Went Right:**

We successfully set up the MySQL database and created the necessary tables (Guests, Events, Sessions, Seats, Bookings) based on the ERD designed in Lucidchart. Writing the SQL commands in MySQL Workbench or a similar client was straightforward once the data model was clear. We were also able to successfully build basic HTML forms in VSCode (as shown in the provided snippet) to capture guest information and allow selection of event/session types, demonstrating the ability to create the user-facing input forms. Using Lucidchart proved very helpful in visualizing the database structure and relationships before writing the actual SQL code, minimizing design errors.

**⦁ What Went Wrong:**

A significant challenge was the difficulty in connecting the static HTML forms built in VSCode directly to the MySQL database. HTML alone cannot process form submissions and interact with a database; a backend programming language (like PHP, Python, Node.js, etc.) was necessary for this, which was beyond the scope of a purely front-end or database-only approach. Implementing the logic to check seat availability per session in real-time based on existing bookings proved complex without a backend processing layer. Importing data from the initial Excel file format was a specific issue that led to errors until it was converted to CSV and the correct LOAD DATA INFILE path was determined. Time constraints prevented the full implementation of features like attendance tracking and generating dynamic reports directly from the web interface. Visually representing the 602 seats and their availability in a user-friendly way within a simple web form also presented a design challenge.

**⦁ Lessons Learned:**

This project strongly reinforced the necessity of a multi-tiered architecture for web applications: a frontend (HTML/CSS via VSCode), a backend (needed for processing logic and database interaction), and a database (MySQL). We learned that effective database design (using tools like Lucidchart) is a critical first step before implementing the tables. We also gained practical experience with debugging SQL errors and understanding the limitations of different technologies (e.g., HTML cannot directly interact with MySQL). Managing project scope and understanding the time required for each component (database setup, frontend design, backend logic) is crucial for successful development.