AFRICA UNIVERSITY
A United Methodist-Related Institution

"Investing in Africa's Future"

COLLEGE OF ENGINEERING AND APPLIED SCIENCES (CEAS)

NCIS 403: WEB APPLICATION DEVELOPMENT

END OF FIRST SEMESTER EXAMINATIONS

NOVEMBER 2025

LECTURER: MS ELIZABETH MAFU

TIME: 3 HOURS

INSTRUCTIONS

1. Answer **all questions**.
2. Use your local development environment (VS Code, Node.js, MongoDB, Browser).
3. Do **not** use the internet. All required tools and libraries are assumed pre-installed.
4. Submit your completed solution folder on the provided flash drive.
5. Ensure your code is **well-commented** and organized.

Question 1 – HTML & CSS (10 Marks)

Create a **responsive landing page** for a fictional service called *"HealthConnect"*. The page must:

- Contain a header with a logo (use placeholder text/logo).
- A navigation menu with at least 3 links.
- A main section with a welcome message and a styled call-to-action button.
- A footer with contact information.
- Use **CSS Grid or Flexbox** to make the layout responsive for both desktop and mobile.

Question 2 – JavaScript (15 Marks)

Enhance the page created in **Question 1** by adding JavaScript functionality:

- A **form** with the following fields: Name, Email, and Message.
- Validate the form so that:
  - All fields are required.
  - Email must be in the correct format (basic regex check allowed).
- Display a success message below the form once it is correctly submitted (no backend required).

Question 3 – React.js (20 Marks)

Create a simple **React application** called *Task Tracker* with the following:

- A form to add new tasks (task name and due date).
- A task list displaying all added tasks.
- A delete button next to each task.
- Use **React hooks** (`useState`) to manage tasks.

Question 4 – Node.js & Express API (20 Marks)

Build a simple **Express REST API** that manages a collection of books. Each book has:

- Title, Author, Year.
  Implement the following routes:
- `GET /books` → returns all books (in JSON).
- `POST /books` → adds a new book.
- `DELETE /books/:id` → deletes a book by ID.

Data can be stored in a simple **in-memory array** (no need for database here).

Question 5 – MongoDB Integration (15 Marks)

Extend **Question 4** to connect to a **local MongoDB database** named `examdb` with a collection `books`.

- Replace the in-memory array with MongoDB queries.
- Implement the same three routes (`GET`, `POST`, `DELETE`).
- Provide at least **one sample document insertion script** (`insertOne`) in your submission.

Question 6 – Authentication (10 Marks)

Add **basic user authentication** to the API in Question 4 or 5.

- A `/login` route that accepts `username` and `password`.
- Hard-code one valid user (e.g., `admin`, password `12345`).
- Only allow access to `POST` and `DELETE` book routes if the user is logged in.

Question 7 – Deployment Simulation (10 Marks)

Without internet deployment, simulate a deployment-ready structure:

- Organize your project into folders (`frontend`, `backend`, `db`).
- Include a `README.md` describing:
  - How to run the backend API.
  - How to run the frontend.
  - Dependencies required.
- Create a simple shell script or batch file that starts both backend and frontend locally.

END OF EXAMINATION