ASSESSMENT AND INTERNAL VERIFICATION FRONT SHEET (Individual Criteria)

Course Title	**Bachelor of Science (Honors) in Software Development / Cybersecurity**	**Lecturer Name & Surname**	James Attard
Unit Number & Title	ITSFT-506-1612 | Server Side Scripting		
Assignment Number, Title / Type	Home Assignment		
Date Set	6h March, 2025	**Deadline Date**	14th March, 2025

Student Name	Lucas Said	**ID Number**	26005H	**Class / Group**	SWD-6.2A

Assessment Criteria	**Maximum Mark**
KU1.1 Show understanding of dynamic web pages and the process of a server side script	*5*
KU1.2 Show understanding of troubleshooting a dynamic web page	*5*
AA1.3 Construct a program to show the difference between client side and server side scripting	*7*
KU1.4 Show understanding of typical features in a framework	*5*
AA1.5 Examine various methods of debugging	*7*
KU2.1 Show understanding of the basic layers of a framework	*5*
KU2.2 Show understanding of various helpers and components available in a framework	*5*
AA2.3 Develop a web application using the features of a server side framework	*7*
SE2.4 Develop and design an application programming interface using the features of server side framework	*10*
AA3.1 Produce relevant code to implement basic controller and model functions	*7*
KU3.2 Show understanding of the validation techniques available in a framework	*5*
SE3.3 Design URLs in a SEO friendly way	*10*
KU4.1 Show understanding of model commands which can be used to interact with data in a database	*5*
AA4.2 Create table associations to be used in an application using the features of a server side framework	*7*
SE4.3 Modify a web application in order to make it safer using the framework's security features	*10*
Total Mark	100

Notes to Students:

- This assignment brief has been approved and released by the Internal Verifier through Classter.
- Assessment marks and feedback by the lecturer will be available online via Classter (Http://mcast.classter.com) following release by the Internal Verifier
- Students submitting their assignment on Moodle/Turnitin will be requested to confirm online the following statements:

Student's declaration prior to handing-in of assignment
- ❖ I certify that the work submitted for this assignment is my own and that I have read and understood the respective Plagiarism Policy

Student's declaration on assessment special arrangements
- ❖ I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit.
- ❖ I declare that I refused the special support offered by the Institute.

College and Student Management System

Objective: Build a Laravel 10-based web application to manage colleges and their students. Demonstrate your understanding of one-to-many relationships, CRUD functionality, filtering, sorting, form validation, and Bootstrap alert messages.

A. Project Requirements

1. **Database Setup**
   - Design two tables:
     - colleges:
       - id (Primary Key, Auto-increment)
       - name (String, required, unique)
       - address (String, required)
       - created_at and updated_at (Timestamps)
     - students:
       - id (Primary Key, Auto-increment)
       - name (String, required)
       - email (String, required, unique)
       - phone (String, required, validated for format)
       - dob (Date of Birth, required, validated)
       - college_id (Foreign Key referencing colleges.id)
       - created_at and updated_at (Timestamps)

2. **Routes**
   - Use resourceful routes for both CollegeController and StudentController:
     - /colleges (index: list all colleges)
     - /colleges/create (create: form to add a new college)
     - /colleges/{id}/edit (edit: form to update a college’s details)
     - /students (index: list all students, with filtering by college)
     - /students/create (create: form to add a new student)
     - /students/{id}/edit (edit: form to update a student’s details)
     - /students/{id} (destroy: delete a student record)

3. **Controllers**
   - Implement CRUD methods in CollegeController and StudentController.
   - Handle filtering logic in the StudentController to show students by selected college.
   - Handle sorting logic in the StudentController to sort students by their name

4. **Blade Views**
   - Use an appropriate Bootstrap or similar framework for styling a master layout file.
   - Implement Blade view templates that should inherit from the Master Layout for the following pages:
     - **Colleges Index:** Displays a list of colleges with options to view, edit or delete.
     - **Students Index:** Displays a list of students with a dropdown filter for colleges, and options to view, edit or delete.
     - **Create/Edit Forms:** Forms for adding or updating students and colleges.

5. **Partial Views**
   - **Filter:** A dropdown component to filter students by college, included on the students' index page.
   - **Sort**: Optional as a partial view, a clickable component that allows students to be sorted by name.
   - **Forms:** Separate partial views for forms used in both create and edit pages.

6. **Validation and Alerts**
   - Validate form inputs:
     - colleges.name should be unique and required.
     - colleges.address should be required.
     - students.name, email, phone, and dob should be required.
     - Validate email for proper format and phone for proper format (e.g., 8 digits).
   - Display Bootstrap alert messages for success or failure during CRUD operations (e.g., “Student added successfully!”).

7. **Eloquent ORM**
   - Define relationships:
     - A college can have multiple students.
     - A student can only be in one college.

B. Instructions

- This is a home-based assessment comprising of a project, a progress report table, a link to publicly accessible GitHub repo, and a link to an unlisted (but not private) 5-min YouTube demo video of your app and code.
- Any form of copying is prohibited. This includes plagiarism, AI tools, etc.
- Late submissions will not be accepted and will be considered as not submitted.
- YouTube videos without a sound or longer than 5 minutes will not be accepted and you may be asked to sit for an interview. Your marks from the YouTube video requirement section will be deducted accordingly.
- The entire project must be version controlled on to a public repository in GitHub. Students must version control each task in to its own branch.
- The final document should be uploaded on VLE. Make sure to choose the correct link for your class. Submissions posted at the wrong link will be considered as not submitted.
- The document should be named in the format class_surname_name (without the dots). For example, SWD62B_Borg_Joseph.
- You may be called for an interview if your lecturer has suspicion of plagiarism, copying or any form of cheating. Interviews will be setup within a few weeks of assignment deadline so it is in the student's best interest to be available. Students not available for the interview to explain their work, will result in an automatic failure.

C. Submission Requirements

1. A ZIP file containing the project folder.
2. A Microsoft Word document (.docx format) including:
   - The Cover Sheet filled with your details, as the first page of this document.
   - Link to the GitHub repository where the entire project is located. The link must be publicly accessible. Ex. https://www.github.com/joeborg/server-side-scripting-project-2025
   - A YouTube link to a screen recording session of your web application, not longer than 5 minutes. You need to demonstrate the working application very briefly (around 1 minute) and the rest of the time to explain the controllers of both Student and College parts. The video is part of the marking and if the explanation is not deemed clear, you may be asked to sit for an interview with your lecturer.
   - A table indicating the progress for each of the requirements in this assignment. **For each task** you need to add a brief comment to describe your completed tasks and for those tasks which are either incomplete or not working properly, you need to also describe your troubleshooting process:

Task	Complete	Incomplete / Not Working	Not Done
Database Setup	*Ex. Created a database called xyz; Added migration files for x and y, etc..*	…	……
Routes	…		
Controllers	…	*Ex, The 'delete' method of the student not working properly due to error xyz.*	*Ex. Did not manage to do the sorting*
Blade Views	…	…	…
Partial views	…	…	…
Validation/Alerts	…	…	…
Eloquent ORM	…	…	…

D. Evaluation Criteria

- Understanding and implementation of a Server Side CRUD application.
- Proper implementation of one-to-many relationships.
- Filtering and sorting functionality on the students' index page.
- Validation of form inputs and proper display of alert messages.
- Clean and organized code with comments.
- Effective use of Master layout, Blade templates and partial views.
- Proper version control of your code.
- Proper self-assessment of your code and understanding of troubleshooting methods (in the form of a tabular progress report).
- Clear presentation and explanation of both your application and the code in the form of a YouTube video.

E. Marking Scheme

- Setup all the database objects, table relationships, database migration files and ORM models. (**AA4.2, 7 marks**)
- Controller setup correctly (**SE2.4, 8 marks**) (**AA3.1, 7 marks**)
- Appropriate Bootstrap layout file with adequate originality integrated correctly with the rest of the views (**AA1.3, 7 marks**)
- Render all dynamic data from database. (**KU1.1, 5 marks**) (**KU1.2, 5 marks**) (**KU2.2, 5 marks**)
- All routes setup correctly. (**SE3.3, 8 marks**)
- Filtering feature setup correctly. (**KU1.4, 5 marks**)
- Sorting feature setup correctly. (**KU4.1, 5 marks**)
- Form validation implemented. (**KU3.2, 5 marks**) (**SE4.3, 8 marks**)
- Editing and Deletion implemented correctly. (**AA2.3, 7 marks**)
- Clear tabular report of your app and code. (**AA1.5, 7 marks**)
- Suitable YouTube video demonstrating the application and explaining clearly the code (**KU2.1, 5 marks**)
- Code version controlled appropriately and frequently on GitHub. (**SE2.4, 2 marks**) (**SE3.3, 2 marks**) (**SE4.3, 2 marks**)