University of Sunderland

Faculty of Business and Technology: Programme -

NAME: Upendra Khanal

STUDENT NUMBER: 240704403

MODULE CODE: CET138

MODULE TITLE: Full Stack Development

MODULE LEADER: David Grey

SUBMISSION DATE AND TIME: 29 AUG 2025/23:59 GMT

ASSIGNMENT:

Academic Misconduct is an offence under university regulations, and this involves:

- **Plagiarism** – where you use information from another information source (including your previously submitted work) and pass it off as your own. This can be through direct copying, poor paraphrasing and/or absence of citations.
- **Collusion** – where you work too closely, intentionally, or unintentionally, with others to produce work that is similar in nature. This can be through loaning of materials, drafts or through unauthorised use of a fellow student's work.
- **Asking another person to write your assignment** – where you ask another individual or company to complete your work for you, be that paid or unpaid, and submit it as if it were your own.
- **Unauthorised use of artificial intelligence** – where you use artificial intelligence tools to generate your assignment instead of completing it yourself and/or where you have not been given permission to use artificial intelligence tools by your module leader. Please complete the following declaration around you use of artificial intelligence tools in your assignment.

STATEMENT ON USE OF ARTIFICIAL INTELLIGENCE TOOLS:

• I have used artificial intelligence tools to generate an idea for my assignment:	**NO**
• I have used artificial intelligence tools to write my assignment for me:	**NO**
• I have used artificial intelligence tools to brainstorm ideas for my assignment:	**NO**
• I have used artificial intelligence tools to correct my original assignment:	**NO**

DECLARATION

- I understand that by submitting this piece of work I am declaring it to be my own work and in compliance with the university regulations on Academic Integrity.
- I confirm that I have done this work myself without external support or inappropriate use of resources.
- I understand that I am only permitted to use artificial intelligence tools in line with guidance provided by my Module Leader, and I have not used artificial intelligence tools outside this remit.
- I confirm that this piece of work has not been submitted for any other assignment at this or another institution prior to this point in time.
- I can confirm that all sources of information, including quotations, have been acknowledged by citing the source in the text, along with producing a full list of the sources used at the end of the assignment.
- I understand that academic misconduct is an offence and can result in formal disciplinary proceedings.
- I understand that by submitting this assignment, I declare myself fit to be able to complete the assignment and I accept the outcome of the assessment as valid and appropriate.

ANY OTHER COMMENTS FROM STUDENT:

What Is Full Stack Development?

Full stack development is a process at its whole building both front end (client-side) as well back end (server-side) web application. It means that a full stack developer can design user interfaces that are engaging and practical and accessible, while also creating secure, efficient, and scalable server-side systems to handle data processing with responses delivered. In today's tech industry, full stack developers are in high demand because they can manage the entire development lifecycle. From gathering a project's requirements and planning its architecture to putting the application live maintaining applications over time (dealing with bugs fixed at any phase). A full stack developer will typically be involved in all aspects of these tasks.

A typical full stack application will involve multiple layers: the front end for user interaction, the back end for business logic and management of information, and the database layer to store retrieve data. By mastering these layers, it is possible to build complete end-to- end solutions which work smoothly as a unit.

Front End (Client-Side)

Definition: The front end refers to everything that a user sees and interacts with directly in their web browser. It ensures that websites and applications display neatly on the user's computer. People who create front ends are responsible for the structure, visual design and interactive behaviour of a website or app.

Core Technologies: The structure of a web page is provided by HTML (HyperText Markup Language). CSS (Cascading Style Sheets) is used to control layout, colors and typography to produce a consistent look throughout the website or application. With JavaScript, it can also make elements that users drag and drop within the page come alive as well as animations become interactive and updates to content occur without a browser reload.

Front-end development also frequently uses frameworks and libraries such as Bootstrap for responsive layouts, Angular or Vue in building single-page applications with complex features.

Responsibilities: Ensuring accessibility for all users, including those with disabilities; laying out responsive designs that work well on mobile, tablet and desktop computers; optimizing performance by minimizing assets and loading resources lazily (not until they are needed but after the final version has been downloaded); and giving users an easy-to-use platform.

Security Considerations: While front-end code is visible to the user, sensitive operations and data validation must be performed on the server. Client-side validation is used to improve user experience but should never replace server-side security checks.

Back End (Server-Side)

Definition: The back end is the part of an application that is hidden from view of end users and run on a server. It houses business logic to process user requests, perform database operations, apply regulations and then send responses back to the front end.

Core Technologies & Patterns: Common back-end environments include Node.js w/ Express, Python with Django and Flask, PHP using Laravel or Java Spring Boot. There are even cases where it might expose APIs that use RESTful Resource API's for communication, also GraphQL which is much more flexible than the former option we mentioned earlier(Http Style). Additionally one could use WebSockets-esque techniques to keep updated real-time data in their app.

Responsibilities: Authenticating and authorizing users, implementing complex workflows, integrating with third-party APIs, managing sessions, and enforcing security policies.

Security Practices: Sanitizing and validating all input to prevent injection attacks, encrypting data in transit with HTTPS/TSL, applying the principle of least privilege for resource access, using secure storage for environment variables and credentials, as well as monitoring any suspicious activity through logging the activities to a database.

Database & Data Layer

Definition: The database layer stores the data it obtains, making sure that any information which has been received is reliably saved, retrived and updated in accordance with the Application program's demands. Access to the database is channeled uniquely through the back end, Keeping it remote from direct access by any user.

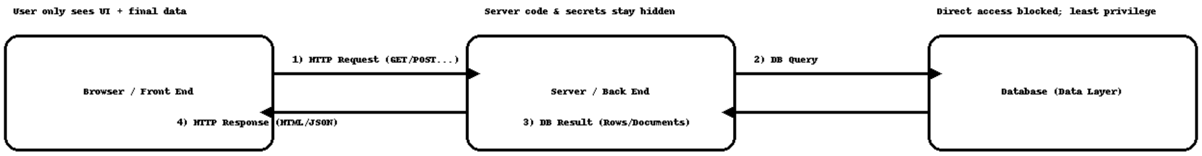
Options: Relational(SQL) databases(example MySQL, PostgreSQL) Just store data in structured tables and enforce schemas with relationships defined in advance. Some more complex queries as well as ACID-compliant transactions are possible for fitting applications. Non-relational(NoSQL) databases like MongoDB and DynamoDB Represent data in flexible ways: documents or key-value pairs for example. Their virtue is quick scaling and easy adaptation to changing demands.

Access & Performance: Back-end applications typically use ORMs (Object-Relational Mappers) like Sequelize, Prisma, or TypeORM for SQL databases, and ODMs (Object-Document Mappers) like Mongoose for MongoDB. Indexing frequently accessed fields, caching results, and optimizing query structure are key to high performance. Schema migrations are used to manage database changes safely.

Security: Enforce least-privilege permissions for database users, encrypt data both at rest and in transit, validate input to avoid malicious queries, and maintain regular, tested backups. Audit logs should be implemented to track changes and access to critical data.

Typical Request–Response Flow (Step-by-Step)

See the diagram below, then read the numbered explanation:


User only sees UI + final data
Server code & secrets stay hidden
Direct access blocked; least privilege
1) HTTP Request (GET/POST...)
2) DB Query
Browser / Front End
Server / Back End
Database (Data Layer)
4) HTTP Response (HTML/JSON)
3) DB Result (Rows/Documents)

1) HTTP Request (Browser → Server): The user triggers an action on the front end (e.g., submitting a form). The browser sends an HTTPS request to the server using GET, POST, PUT, or DELETE, including any necessary headers, cookies, and payloads.

2) Database Query (Server → Database): The server authenticates the user, validates the request data, applies business logic, and issues a parameterized query to the database through an ORM/ODM or direct driver.

3) Database Result (Database → Server): The database processes the query and returns the requested records or a confirmation. The server processes these results and removes sensitive information before preparing the response.

4) HTTP Response (Server → Browser): The server sends the final response (HTML, JSON, etc.) back to the browser. The front end then updates the interface accordingly. At no point is the server-side code or database directly exposed to the user.