National University
Of Computer and Emerging Sciences

System Architecture

An Assignment presented to

Sir Basharat Hussain

In partial fulfillment
of the requirement for the course of

Software Engineering

By

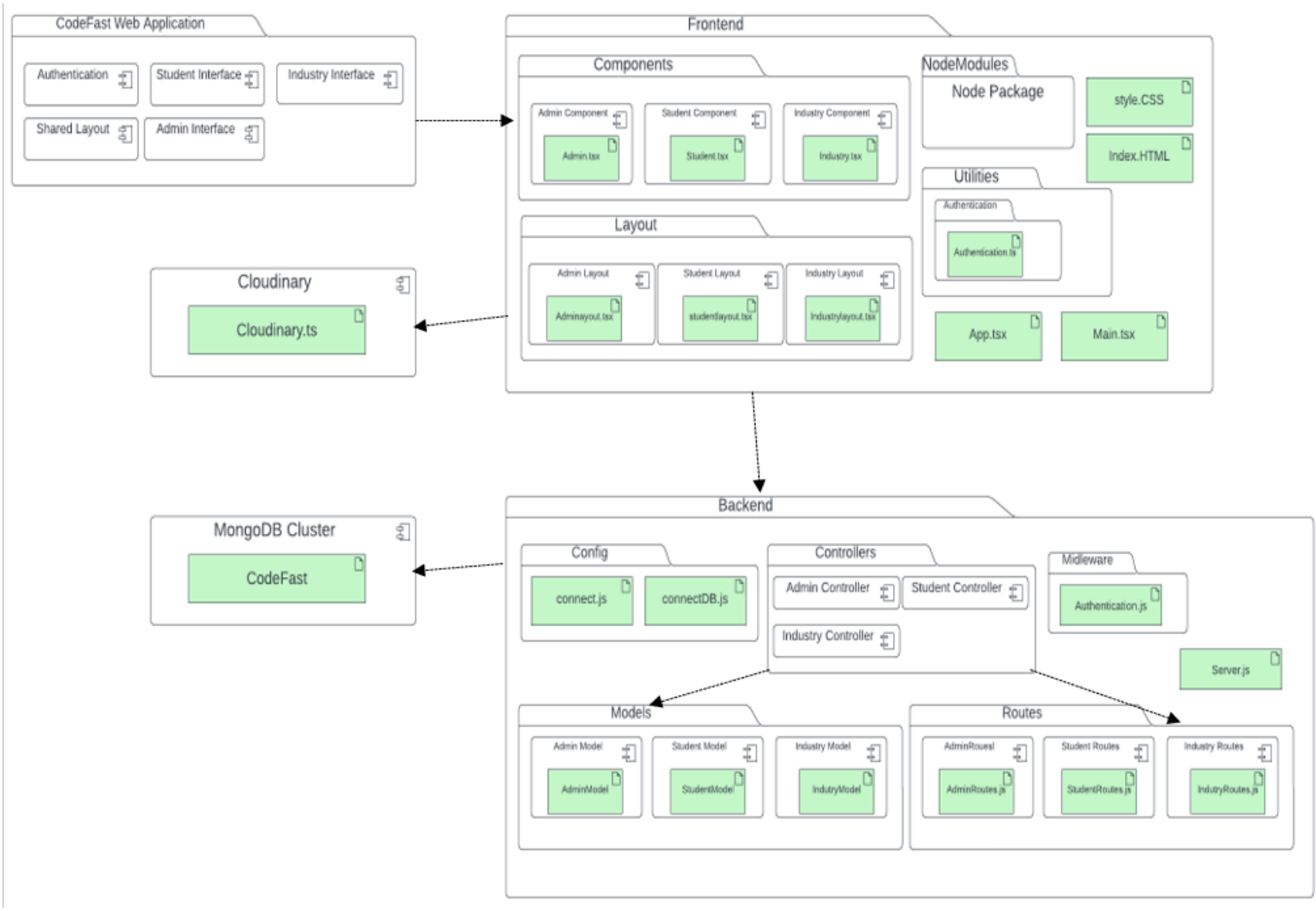
Arban Arfan(22I-0981), Zakariya Abbas (22I-0801), Messam Raza (22I-1194)

BS(CS)
SECTION-E

Identifying Subsystems

Package Diagram

In this section, we present the decomposition of the CodeFast Web Application into its major subsystems, as represented in the UML Package Diagram. The purpose of this identification is to modularize the system into logically coherent groups to improve maintainability, scalability, and clarity of the system architecture.



Subsystem Overview

The CodeFast Web Application is organized into the following major subsystems:

---

1. Frontend Subsystem

The *Frontend* subsystem is responsible for the user interface and client-side logic. It is further subdivided into:

- **Components Package:**
  Contains reusable and specialized components for different user types, including Admin, Student, and Industry components.
- **Layout Package:**
  Manages the structural layouts for Admin, Student, and Industry interfaces, ensuring consistent user experience across different sections.
- **Utilities Package:**
  Provides shared functionality such as Authentication utilities and core configuration files
- **NodeModules Package:**
  Incorporates third-party libraries and assets such as `style.css` and `index.html`, essential for styling and initialization.

The frontend subsystem interacts with external services (e.g., Cloudinary) and communicates with the backend server.

---

2. Backend Subsystem

The *Backend* subsystem handles business logic, database operations, API endpoints, and server configuration. It is divided into:

- **Config Package:**
  Includes database configuration files (`connect.js`, `connectDB.js`) necessary for establishing connections with the MongoDB database.
- **Controllers Package:**
  Defines the core controllers responsible for managing operations for Admin, Student, and Industry entities.
- **Models Package:**
  Contains the Mongoose models for Admin, Student, and Industry, which define the schema and structure for database interactions.
- **Routes Package:**
  Exposes API endpoints via Express.js routing modules for Admin, Student, and Industry operations.

- **Middleware Package:**
  Provides authentication middleware to safeguard protected routes and ensure authorized access to resources.

The backend subsystem interacts with both the *MongoDB Cluster* and the *Cloudinary Service* for persistent storage and media management, respectively.

---

3. Cloudinary Service Subsystem

The *Cloudinary Service* subsystem is an external service integration that manages image and file uploads. The frontend and backend subsystems depend on this service for efficient media storage and retrieval, enhancing the application's capabilities for handling user-uploaded assets.

---

4. MongoDB Cluster Subsystem

The *MongoDB Cluster* subsystem provides the centralized database solution for the CodeFast Web Application. It stores, retrieves, and manages all persistent data related to Admins, Students, and Industries. The backend subsystem is directly dependent on this database cluster for its operations.

Subsystem Interactions

- The *Frontend* subsystem depends on the *Backend* subsystem for data fetching and user management.
- Both *Frontend* and *Backend* subsystems interact with the *Cloudinary Service* for media management.
- The *Backend* subsystem communicates with the *MongoDB Cluster* for persistent data storage and retrieval.

Architecture Styles

1. Client-Server Architecture

The **CodeFast** application primarily follows a **Client-Server Architecture** model.
In this architecture, the frontend (client) and backend (server) are logically separated:

- **Client (Frontend):**
  Developed using **React** and **TypeScript**, the client side is responsible for rendering user interfaces, managing user interactions, and making asynchronous requests to the server for data.
- **Server (Backend):**
  Developed using a backend framework that follows the **Model-View-Controller (MVC)** design pattern, the server processes business logic, manages data persistence, and responds to client requests with appropriate data or actions.

This separation ensures **scalability**, **maintainability**, and **flexibility** in deployment across distributed environments.

---

2. Frontend Architectural Style: Modular Feature-Based Architecture

The frontend design of CodeFast adheres to a **Modular Feature-Based Architecture**.
Key characteristics include:

- **Component-Based Structure:**
  The frontend is organized into domain-specific folders (`Admin`, `Auth`, `Industry`, `User`, `Shared`), promoting high cohesion within features and low coupling between them.
- **Separation of Layouts and Utilities:**
  Layout components (`AdminLayout`, `IndustryLayout`, `StudentLayout`) and utility functions (such as authentication guards and API instance handlers) are isolated in their respective directories, enhancing code reuse and separation of concerns.
- **Scalable Design:**
  By modularizing based on features, the application can easily scale to accommodate new modules or features without disrupting the existing codebase.

This architectural choice results in a **clean**, **manageable**, and **extensible** frontend structure.

---

3. Backend Architectural Style: Model-View-Controller (MVC)

The backend of CodeFast is structured using the **Model-View-Controller (MVC)** architectural pattern.
It is composed of the following layers:

- **Model:**
  Represents the data layer and business logic, interacting with the database and defining the data structure.
- **View:**
  In the context of an API-driven backend, the "View" corresponds to the API responses sent to the client, typically in JSON format.
- **Controller:**
  Manages the flow between the Model and the View. It handles incoming HTTP requests, invokes appropriate services or models, and returns responses to the client.

This structure enforces a **clear separation of concerns**, facilitating **testability**, **readability**, and **ease of maintenance**.

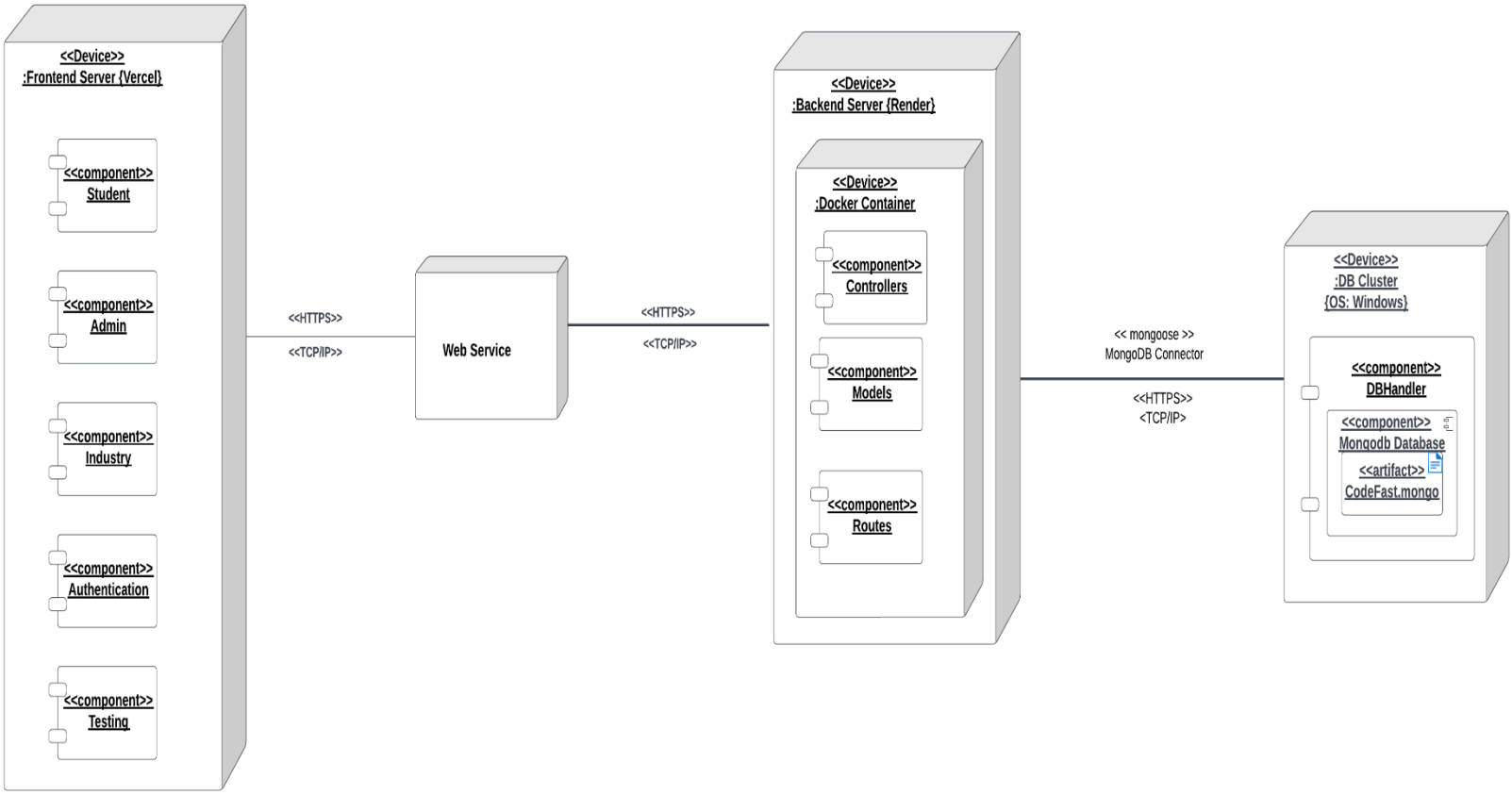
---

4. Additional Design Principles

- **Reusability:**
  Common functionalities and components are abstracted into shared modules and utility functions, ensuring minimal redundancy.
- **Loose Coupling and High Cohesion:**
  Modules are independently developed and tested, leading to improved flexibility and easier future enhancements.
- **Consistency in Code Structure:**
  Adherence to consistent naming conventions, file organization, and interface designs promotes a professional and scalable codebase.

By integrating multiple architectural styles — **Client-Server Architecture** for system communication, **Modular Feature-Based Architecture** for frontend structure, and **Model-View-Controller (MVC)** for backend organization — CodeFast achieves a highly **organized**, **scalable**, and **maintainable** web application architecture, aligning with modern software engineering best practices.

Deployment Diagram



Component Diagram