Course Project Report Structure

This structure is designed to apply to any of the four projects (example: To-Do List, Grade Management, URL Shortener, Chat App) and ensures all technical requirements are met.

Section	Focus	Purpose
1. Cover Page & Info	Metadata	Title, Student Info, Course Name, Project Name.
2. Executive Summary	High-Level Overview	Briefly state the project's goal, the architectural pattern used, and the final achievement.
3. Project Requirements & Goals	Functional Context	List the core **Functional Requirements** (e.g., CRUD, Redirection, Real-Time Messaging) and key **Quality Attributes** targeted (e.g., High Read Performance, Real-Time Response).
4. Architectural Design & Implementation	Core Technical Execution	Detail the technical stack, data models, and the implementation of core logic. **(Crucial Section)**
5. Testing & Verification	Proof of Functionality	Document tests demonstrating that the functional requirements and quality goals were met.
6. Conclusion & Reflection	Learning Summary	Summarize lessons learned and potential future improvements (e.g., adding security, deployment).

Required Content Breakdown by Project

The following table specifies the necessary deliverables for Section 4 (Design & Implementation) and Section 5 (Testing & Verification) for each of the four projects.

Project	Section 4: Design & Implementation (Artifacts/Code Snippets)	Section 5: Testing & Verification (Proof)
1. To-Do List App	* **Architecture:** 1-Tier/Monolithic Structure (Diagram). * **Model:** Python list/dictionary structure for a Task. * **Code:** Snippets for the **CRUD** endpoints (POST to create, PUT to update completion).	* **Verification: cURL/Postman** outputs showing successful creation (HTTP 201) and successful deletion (HTTP 204) of a task.

Project	Section 4: Design & Implementation (Artifacts/Code Snippets)	Section 5: Testing & Verification (Proof)
2. Grade Mgmt System	* **Architecture:** 2-Tier/Layered Structure (Client-Server, Database). * **Model:** SQLAlchemy Schemas for Student, Course, and Enrollment tables. * **Code:** Python function/route implementing the **GPA calculation logic** (weighted average).	* **Verification: cURL/Postman** output retrieving a student profile and showing the calculated, verified **GPA** (e.g., 3.43 for Student 1).
3. URL Shortener Service	* **Architecture:** 3-Tier/High-Read Performance Focus. * **Data Store:** Explanation of the dual-store approach (**SQLite for mapping, Redis for cache/clicks**). * **Code:** The Python function for **Base62 Encoding** (encode_id). * **Code:** The route handling the **HTTP 302 Redirect**.	* **Verification:** Output showing a **cURL** request receiving a **302 Redirect** status. * **Proof: Redis CLI** output (GET clicks:<code>) confirming the **click count** was incremented after the redirect.
4. Real-Time Chat App	* **Architecture: Event-Driven Architecture (EDA)** via **WebSockets**. * **Code (Server):** Node.js/Socket.IO logic for handling the connection and **io.emit('chat message', ...) broadcast**. * **Code (Client):** JavaScript snippet showing the **socket.on('chat message', ...)** listener updating the DOM.	* **Verification:** Screenshot showing **two separate browser tabs** with two different usernames, demonstrating that a message sent in one tab appears **instantly** in the other tab.