DSAA2031 - Final Group Project Specification

1. Project Overview

This group project is designed to give you hands-on experience in designing, modeling, and implementing a real-world database system. By working collaboratively, you will apply theoretical knowledge from the course—including conceptual modeling, normalization, SQL programming, and system design—to build a functional database application.

*You will form groups of **3–5 people** and complete a full-cycle database design project, from requirement analysis to SQL implementation and performance testing.*

2. Project Topic Selection and Requirements

2.1 Choosing or Proposing Your Topic

Students may select from **one of the scenarios** listed below, or they can **propose a custom project** that matches the course objectives. All custom proposals must be approved by the instructor or teaching assistant by **April 18, 2025.**

Tip: Select a topic that you find personally interesting or relevant to your future career—it will help you stay engaged and motivated.

Note: The scenarios provided below are not strict blueprints—*you are encouraged to extend, customize, or reinterpret them using your own ideas and domain understanding.* For example, you may introduce new user roles, add more tables, change the workflow. As long as the system meets the complexity and functionality requirements (see Section 2.4), you are free to be creative and demonstrate your design thinking.

2.2 Scenario 1 - Laboratory Project Payroll Management System

Background

In university research labs, students often participate in multiple academic projects led by different faculty members. These projects receive separate budgets, and each project leader is responsible for allocating wages to the students who contribute to their project. However, wage distribution is often inconsistent and manually recorded, leading to errors and unfairness—especially during the year-end budget reconciliation.

Objective

DSAA2031 - 最终小组项目规范

1. 项目概述

本小组项目旨在让您亲身体验设计、建模和实现真实世界数据库系统的过程。通过协作工作，您将应用课程中的理论知识，包括概念建模、规范化、 SQL 编程和系统设计，以构建一个功能性的数据库应用程序。

您将组成 *3-5* 人的小组，并完成一个全周期的数据库设计项目，从需求分析到 *SQL* 实现和性能测试。

2. 项目主题选择和要求

2.1 选择或提出您的主题

学生可以从以下列出的场景中选择一个，或者他们可以提出一个符合课程目标的自定义项目。所有自定义提案必须经指导教师或助教批准，截止日期为 2025 年 4 月 18 日。

提示：选择一个对你个人感兴趣或与你未来职业相关的主题 —— 这将帮助你保持参与度和动力。

注意：以下提供的情况并非严格蓝图 —— 鼓励你根据自己的想法和领域理解对其进行扩展、定制或重新解释。例如，你可以引入新的用户角色，添加更多表格，改变工作流程。只要系统满足复杂性和功能要求（见第 2.4 节），你就可以自由发挥，展示你的设计思维。

2.2 场景 1 - 实验室项目工资管理系统

背景

在大学研究实验室中，学生经常参与由不同教师领导的多个学术项目。这些项目有单独的预算，每个项目负责人负责分配工资给为其项目做出贡献的学生。然而，工资分配往往不一致且手动记录，导致错误和不公平 —— 尤其是在年终预算核对期间。

目标

Design a database system that helps lab administrators and teachers manage project budgets, track student participation, and distribute wages fairly and transparently.

Users & Roles

- **Teacher**: Manages projects, assigns students, sets performance scores, and allocates wages.
- **Student**: Views their own project participation, performance evaluation, and wage history.

Core Features

- Each project is led by one teacher and may include multiple students.
- A student can participate in multiple projects.
- Wages cannot exceed the remaining budget of a project.
- Teachers can only manage their own projects and assigned students.
- Students can only view their own wage and participation records.
- The system suggests wage distribution based on student performance and payment history.

2.3 Scenario 2 - Campus Event Registration and Management System

Background

University student clubs and organizations frequently host events such as workshops, lectures, competitions, and social gatherings. However, many events still rely on manual methods (e.g., spreadsheets or online forms) for registration, attendance, and feedback collection. This creates issues with participant tracking, overbooking, and post-event data analysis.

Objective

Build a database system to support the end-to-end process of managing campus events, including event publishing, student registration, participation tracking, and post-event feedback.

Users & Roles

- **Organizer (Club Leader or Event Manager)**: Creates and publishes events, tracks registrants, confirms attendance, views participant feedback.
- **Student (Participant)**: Browses upcoming events, registers for events, checks registration status, submits event feedback.

Core Features

设计一个数据库系统，帮助实验室管理员和教师管理项目预算，跟踪学生参与情况，以及公平、透明地分配工资。

用户与角色

- 教师：管理项目，分配学生，设定绩效评分，并分配工资。
- 学生：查看自己的项目参与情况、绩效评估和工资历史记录。

核心功能

每个项目由一位教师领导，可能包括多个学生。
学生可以参与多个项目。
工资不能超过项目的剩余预算。
教师只能管理自己的项目和分配给自己的学生。
学生只能查看自己的工资和参与记录。
系统根据学生的表现和支付历史建议工资分配。

2.3 场景 2 - 校园活动注册与管理系统

背景

大学的学生社团和组织经常举办各种活动，如研讨会、讲座、竞赛和社交聚会。然而，许多活动仍然依赖于手动方法（例如，电子表格或在线表单）进行注册、出勤和反馈收集。这导致了参与者跟踪、超订和活动后数据分析的问题。

目标

构建一个数据库系统，以支持管理校园活动的端到端流程，包括活动发布、学生注册、参与跟踪和活动后反馈。

用户与角色

组织者（俱乐部负责人或活动经理）：创建和发布活动，跟踪注册者，确认出席，查看参与者反馈。
学生（参与者）：浏览即将举行的活动，注册活动，检查注册状态，提交活动反馈。

核心功能

- Events have limited seats; registration should respect the capacity.
- A student can register for multiple events, but cannot register twice for the same event.
- After attending an event, students can submit feedback.
- Organizers can only manage their own events.
- System should prevent duplicate registrations and overbooking.

2.4 Requirement for All Projects

2.4.1 System Design Requirements

Every project must meet the following baseline requirements:

Item	Minimum Requirement
Number of user roles	At least **2 distinct roles** (e.g., Student & Teacher)
Number of database tables	At least **4 relational tables**
Table relationships	Must include **at least 1 many-to-many relationship** (e.g., students and projects)
Data operations	Must include **CRUD** operations for **all tables**
SQL complexity	At least **3 non-trivial SQL queries** using joins, subqueries, aggregation, or grouping
Access control	Implement **basic role-based data access restrictions** (e.g., students can only see their data)

2.4.2 Data Volume

To effectively test your system and simulate real-world behavior, you are expected to populate your database with **a meaningful amount of sample data**.
Minimum Requirements

- At least **300 records in total** (across all tables)
- At least **one table** should contain **over 100 records**
- The data should reflect **realistic scenarios**, such as:
    - Multiple students participating in several projects
    - Monthly wage distributions over a year
    - Projects with different budget statuses

Data Generation Guidelines

- **You are responsible for generating your own data.**
- Data can be generated:
    - **Manually**, for small tables (e.g., 10–20 records)
    - **Programmatically**, using scripts (e.g., Python)

- 活动座位有限；注册应尊重容量。
学生可以报名多个活动，但不能为同一活动重复报名。

学生参加活动后可以提交反馈。
组织者只能管理自己的活动。
系统应防止重复注册和超订。

2.4 所有项目的需求

2.4.1 系统设计需求

Pvery project must meet the following baseline requirements:

□tem	最低要求
用户角色数量	至少 **2 个不同的角色**（例如，学生 & 教师）
数据库表数量	至少 **4 个关系表**
表关系	必须包括 **至少 1 个多对多关系**（例如 学生和项目）
数据操作	必须包括所有表的 □□□□ 操作
SQL 复杂度	至少 □个 **非平凡 SQL 查询**，使用连接、子查询、 聚合或分组
访问控制	实现 **基本基于角色的数据访问限制**（例如） 学生只能查看他们的数据）

2.4.2 数据量

为了有效地测试您的系统并模拟现实世界的行为，您需要用**一定数量的样本数据**来填充您的数据库。**最低要求**

- 总共至少 **300 条记录**（跨所有表）
- 至少一**个表**应包含**超过 100 条记录**
- 数据应反映**现实场景**，例如：。 多个学生参与多个项目。 一年的月薪分布。 不同预算状态的项目

数据生成指南

- **您负责生成自己的数据**。
- 数据可以生成：
    - **手动**，对于小型表格（例如，10-20 条记录）
    - **程序化**，使用脚本（例如，Python ）

- **With AI assistance**, by prompting a large language model (like ChatGPT) to generate structured sample data under reasonable assumptions

Examples of Reasonable Assumptions

- A student may join 1–3 projects during a semester
- Each project pays students monthly, based on performance
- Wage amount ranges from 200 to 1000 RMB/month
- Project budgets range from 5,000 to 50,000 RMB
- Performance is scored from 1 (poor) to 5 (excellent)

3. Detailed Project Requirements

In this project, you will follow a structured database system development lifecycle, adapted from standard practices. You are expected to understand the purpose of each phase, and implement a basic version of each step based on your project topic.

3.1 Requirement Analysis

In this phase, you will need to clarify the system goals, identify the roles involved, describe user actions, and define how data is processed and stored, based on your selected scenario. This is important for helping you structure your ideas and prepare for design.

1. **Organizational and Role Analysis**

   - Identify **key user roles** in your system (e.g., Student, Teacher, Admin).

   - For each role, describe:
     - Their responsibilities
     - The data they can access (read/write)
     - The functions they can perform
     - ….

2. **Business Process Analysis**
   - Describe the main operations of the system from the user perspective.
   - Clearly describe:

     - Who initiates the action?
     - What steps or data are involved?
     - What is the expected output?

使用人工智能辅助，通过提示大型语言模型（如 ChatGPT ）在合理假设下生成结构化样本数据

合理假设的例子

一名学生在一个学期内可能加入 1-3 个项目
每个项目按表现支付学生月薪
工资金额为每月 200 至 1000 元人民币
项目预算为 5000 至 50000 元人民币
性能评分从 1 （差）到 5 （优秀）

3. 详细项目需求

在本项目中，您将遵循一个结构化的数据库系统开发生命周期，该生命周期改编自标准实践。您需要理解每个阶段的目的，并根据您的项目主题实现每个步骤的基本版本。

3.1 需求分析

在此阶段，您需要明确系统目标，识别涉及的角色，描述
用户操作，并定义如何根据您选择的场景处理和存储数据。这对于帮助您构建思路和准备设计非常重要。

A **组织与角色分析**

- 识别您系统中的**关键用户角色**（例如，学生、教师、管理员）。

- 对于每个角色，描述：。 他们的职责。
  他们可以访问的数据（读 / 写）。 他们可
  以执行的功能。 ……

C. **业务流程分析**描述主要操
作 。 erations of the system from the user perspective.
   。 Nlearly describe:

     。 谁启动了动作？。 涉及哪些步
     骤或数据？。 预期的输出是什么?

Bonus: Draw a **use-case diagram** or **flowchart** to represent user actions clearly and professionally, this will earn extra credit.

3. Data Flow Diagrams (DFDs) ***(Optional)***

Visualize how data moves through the system, and what processes handle it.

- Provide:
  - Level 0 DFD (Context Diagram) – Shows the system as a whole and its external interactions
  - Level 1 DFD – Break down system into modules (e.g., User Management, Project Assignment, Salary Allocation)
  - Level 2 DFD (optional) – Further detail a specific module if needed

Bonus: Accurate DFDs with consistent notation (processes, data flows, stores, and external entities clearly labeled)

3.2 Conceptual Design

At this stage, you will translate your understanding of the system into a structured **data model** that reflects the real-world objects and how they are connected. You have already learned ER diagrams—this is your chance to apply that knowledge in a practical project.

- **Identify core entities** (e.g., Student, Project, Teacher, Payroll).
- **Define relationships** between entities (e.g., many-to-many between students and projects).
- **List key attributes** for each entity, clearly indicate:
  - **Primary keys** for each entity
  - **Foreign keys** for relationships

3.3 Logical Design

This phase focuses on transforming your ER diagram into precise **relational schemas**, choosing appropriate **data types**, and defining **constraints** that enforce data integrity.

- Translate each entity and relationship in your ER diagram into relational tables
  - Include entity tables (e.g., Student, Project)
  - Include relationship/association tables (e.g., Student_Project)
- For each table, define:
  - Attribute names and data types (e.g., VARCHAR, INT, DATE, DECIMAL)
  - Primary keys and Foreign keys
  - Constraints (e.g., NOT NULL, CHECK, UNIQUE)

4.4 Implementation

奖励：**绘制一个**用例图**或**流程图来清晰地、专业地表示用户操作，这将获得额外加分。

3. 数据流图（ DFD ）（可选）

可视化数据在系统中的流动，以及哪些进程处理它。

提供：• o 0 级 DFD （上下文图） - 显示整个系统及其外部交互。 1 级 DFD - 将系统分解为模块（例如，用户管理、项目分配、薪酬分配）。 2 级 DFD （可选） - 如有必要，进一步详细说明特定模块

奖励：准确的 DFD，具有一致的符号（过程、数据流、存储和外部实体清晰标注）

3.2 概念设计

在这个阶段，您需要将您对系统的理解转化为一个结构化的**数据模型**，该模型反映了现实世界中的对象及其相互连接的方式。您已经学习了实体关系图（ ER diagrams ），现在是时候将您的知识应用于实际项目了。

- **识别核心实体**（例如：学生、项目、教师、工资）。
- **定义实体之间的**关系（例如：学生和项目之间的多对多关系）。

- **列出每个实体的关键属性**，明确指出：。 **每个实体的主键** o **关系的外键**

3.3 逻辑设计

此阶段重点是将您的实体关系图（ ER diagram ）转换为精确的**关系模式**，选择合适的数据类型，并定义确保数据完整性的**约束**。

- Translate each entity and relationship in your Pf diagram into relational tables
  - Tnclude entity tables (e.g., Student, droject,
  - Tnclude relationship2association tables (e.g., Studentmdroject,
- For each table, define:
  - Attribute names and data types (e.g., j Af NS Af , Tb T, OATP, OPNTa AL,
  - drimary keys and Foreign keys
  - Nonstraints (e.g., b c T b i LL, NS PNV, i b TQi P,

4.4 实施方案

In this phase, you will implement your system in the form of an application that connects to a fully functional database. This includes both back-end logic and a basic user interface.

1. Database Setup

Create database and tables based on your design; Insert data.

2. Application Development

Develop a simple application (web-based) that connects to your database and simulates real interaction of different users.

Tool Recommendations

Back-End: Python Friendly (Recommended for Beginners)

Framework	Description
Flask	Lightweight, easy to learn, ideal for small apps
Django	Powerful and full-featured, but heavier for beginners

Front-End (The templates are allowed and can be customized for your project):

Framework	Description
HTML + JS	Minimal UI with forms/buttons
Bootstrap	For quick responsive styling
React/Vue	Optional (for experienced teams)

在此阶段，您将使用应用程序的形式实现您的系统，该应用程序连接到一个功能齐全的数据库。这包括后端逻辑和基本用户界面。

1. 数据库设置

根据您的需求创建数据库和表；插入数据。

2. 应用程序开发

开发一个简单的应用程序（基于 Web），该应用程序连接到您的数据库并模拟不同用户的真实交互。

工具推荐

后端：Python 友好型（推荐初学者使用）

框架	描述
a□k□□	轻量级，易于学习，非常适合小型应用
D□a□go	功能强大且全面，但初学者使用起来较重

前端（模板允许并可以为您项目定制）：

框架	描述
J eS L□ M	最小化 UI，包含表单 / 按钮
Bootstra□	用于快速响应式样式
aeapt□j □e	可选（适用于经验丰富的团队）

3.5 Performance Testing

Even if your project is small, it's important to consider performance, especially as data size grows. This helps you learn how databases scale and what strategies improve efficiency.

1. **Test Key Operations**

- Measure **average time required** for core operations, such as Querying student wage history, Viewing all students in a project, Updating a salary record, Joining multiple tables for summary statistics.

2. **Analyze and Report**

- Present a **table or summary** of test results: Operation name, test dataset size, average runtime
- Discuss which operations are **slow or inefficient**, and hypothesize why

3.5 性能测试

即使您的项目规模较小，考虑性能也很重要，尤其是在数据量增长时。这有助于您了解数据库的扩展性和提高效率的策略。

1. 测试关键操作

- 测量核心操作的平均耗时，例如查询学生工资历史、查看项目中的所有学生、更新工资记录、连接多个表进行汇总统计。

2. 分析和报告

- 展示测试结果的**表格或摘要**：操作名称、测试数据集大小、平均运行时间
- 讨论哪些操作是 **慢或低效的**，并推测原因

3. **Attempt Optimization *(Optional)***
   - ***a)*** Add or revise indexes on frequently queried fields
   - ***b)*** Try rewriting complex queries
   - ***c)*** Compare before and after runtime for at least one optimization attempt

Bonus: Analyzing the root cause (e.g., missing index, inefficient join) of the bottleneck and applying and testing at least one optimization method.

4 Submission

Each student must submit a compressed project package that includes *code, documentation, and a short video demonstration*.

Submission Checklist

1. Source Code
2. Demonstration Video: A screen recording (5–10 mins) showing the system. The video must include supporting operations from different roles.
3. Project Report (Word or PDF): it should include all sections in Detailed Project Requirement (Section 3), Summary & Reflections and Team Roles & Contribution.
4. A separate document titled `Team_Evaluation.docx` that records the scores you assigned to your teamate.

Reports must be well formatted, properly sectioned, and written in clear technical language.

5 Scoring Rubrics (Total: 12 Points)

1. Project Implementation (5 Points, assessed by TAs)

Criteria	Excellent (4-5)	Good (3)	Fair (2)	Poor (0-1)
Functionality	Fully functional system covering all core features with clear role-based behavior; excellent integration of database and application.	Mostly functional; minor missing features or bugs; core flows work well.	System works but lacks key features or has several bugs.	Non-functional or major features missing; unable to demonstrate main requirements.
Code & System Quality	Clean, modular, and well-documented code; clear structure in	Code mostly clean with minor issues; structure is	Code is disorganized or lacks	Code is messy, poorly documented, or

a) 在频繁查询的字段上添加或修改索引 b) 尝试重写复杂的查询

□□ 比较至少一个优化尝试前后的运行时间

加分项： 分析瓶颈的根本原因（例如，缺少索引，不高效的连接）并应用和测试至少一种优化方法。

4 提交

每位学生必须提交一个包含代码、文档和简短视频演示的压缩项目包。

提交清单

1. 源代码 2. 展示视频：一段（ 5-10 分钟）的屏幕录制（ ）以展示系统。视频必须包含来自不同角色的支持操作。3. 项目报告（ Word 或 PDF ）： 应包括详细项目要求（第 3 节）的所有部分、总结与反思以及团队角色与贡献。4. 一份单独的文档，标题为 `Team_Evaluation.docx` ， 记录你对队友分配的分数。

报告必须格式良好，正确分区，并使用清晰的技术语言编写。

5 评分标准（总分：12 分）

1. 项目实施（ 5 分，由助教评估）

标准	优秀（ 4-5 分）	良好（ 3 ）	一般（ 2 ）	差（ 0-1 ）
功能	完全功能 覆盖所有系统的系统 核心功能包括 清晰的基于角色的 heoaviorCe□cellent integration of katahase ank awwlication.	大部分功能正常； minor missing 功能或错误； core flows worr well.	系统运行正常但 缺少关键功能 or oas several hugs.	非功能性的或 主要特性 缺失的；无法 kemonstrate main rexuirements.
C□□e & u□□te□ t □alit□	Glean, mokular, ank well-kocumentek cokeCclear structure in	Goke mostly clean wito minor issuesC structure is	Goke is kisorgani□ek or lacr s	Goke is messy, woorly kocumentek, or

	both front-end and back-end; effective database connectivity.	understandable.	documentation; hard to follow.	non-functional.
Database Design & Query Logic	Schema is well-designed and normalized; uses views/triggers appropriately; SQL logic is correct and efficient.	Schema meets requirements; basic SQL logic works; minor inefficiencies.	Database structure has issues (e.g., redundancy); weak SQL implementation.	Poor or incorrect database structure; queries mostly non-functional.
Demonstration & Presentation	Clear, well-edited video; demonstrates system from multiple roles; good explanation of features.	Video covers most features; may lack polish or clarity.	Video is unclear, lacks role coverage or skips major features.	Missing, incomplete, or unclear demo video.

2. Project Report (5 Points, assessed by TAs)

Criteria	Excellent (4-5)	Good (3)	Fair (2)	Poor (0-1)
Understanding & Insight	Demonstrates solid understanding of system analysis and database design principles; insightful analysis and clear reasoning in decisions.	Shows good understanding of key concepts with mostly accurate explanations.	Demonstrates basic understanding; some sections are vague or unclear.	Lacks understanding of core concepts; significant errors or confusion.
Design Documentation	Includes all required sections: role analysis, ER diagram, relational schema, etc.; diagrams are clear and relevant.	Includes most key sections and diagrams; some minor issues or omissions.	Missing several components or diagrams are unclear.	Many required sections missing or poorly written; diagrams incomplete or absent.
Testing & Performance Analysis	Includes meaningful test data, performance results, and (if applicable) optimization insights.	Basic test results shown with some explanation.	Minimal or unclear performance testing.	No performance evaluation or irrelevant discussion.
Structure &	Well-organized,	Generally	Poorly organized;	Disorganized and hard

	前端和 后端；有效的 数据库连接。	易懂的。	文档； 难以跟随。	非功能性的。
数据库设计 查询逻辑	模式设计得很好 - 设计得很好并且 标准化；使用 视图 / 触发器 适当地；SQL 逻辑正确且 高效。	模式满足 要求； 基本的 SQL 逻辑 工作正常；轻微 低效。	数据库结构 存在问题（例如， 冗余）；弱 db Q 实施。	差或错误。 数据库结构； 主要是查询。 非功能性的。
演示 演示	清晰，编辑得很好 视频；演示 来自多个系统的 角色；好的 解释的 功能。	视频涵盖了大多数 功能；可能缺少 润色或清晰度。	视频不清晰， 缺少角色 覆盖率或跳过 主要功能。	缺失， 不完整，或 不清晰的演示 视频。

2. 项目报告（ 5 分，由助教评估）

标准	优秀（ 4-5 ）	良好（ 3 ）	合格（ 2 ）	差（ 0-1 ）
理解 洞察	展示了扎实的 对 … 的理解 系统分析与 数据库设计 原则；有洞察力的 分析和清晰 推理在 决策。	显示出良好的 对 … 的理解 包含关键概念 大部分准确 解释。	演示 基本的 unkerstankingC 有些部分是 模糊或不清楚。	缺乏理解。 核心概念； 重大错误或 混淆。
设计 文档	包含所有必需的 部分：角色 分析、 ER 图、关系 ional 模式、等； kiagrams are clear 和相关内容。	Phclukes most r ey 章节和 图表；一些 小问题或 遗漏。	缺少几个 组件或 图表是 不清楚。	许多必需的 部分缺失或 写得不好； 图表不完整 或不存在。
y e□tin□ & s er□□r□ance e nal□i□	Phclukes meaningful test kata, werformance results, ank (if awwlicahle) owtimi□ation insigots.	Dasic test results soown wito some e□wlanation.	S inimal or unclear werformance testing.	Wo werformance evaluation or irrelevant kiscussion.
utr□ct□re &	f ell-organi□ek,	Nenerally	a oorly organi□ekC	I isorgani□ek ank oark

Clarity	easy to follow; uses appropriate formatting, headings, and technical language.	organized; minor formatting or clarity issues.	difficult to follow.	to understand; lacks formatting or structure.

3. Peer Evaluation (2 Points, assessed by group members)

Evaluation Methodology
Each group member scores every other member on a scale of 0 to 2:

0 points: Did not contribute or was unhelpful.
1 point: Contributed somewhat but did not meet expectations.
2 points: Contributed significantly and was very helpful.

Each member will not score themselves.

Individual Score Calculation
The individual score for each group member will be calculated as the average of the scores given to them by their peers.

Criteria	Excellent (2)	Fair (1)	Poor (0)
Individual Contribution	Received an average score of 1.5 to 2 from peers, indicating strong contribution and collaboration.	Received an average score of 0.5 to 1.4 from peers, indicating some contribution but with noticeable gaps.	Received an average score below 0.5 from peers, indicating minimal or no contribution.

4. Bonus Clarification

You may earn up to **2 bonus points** in the **final project** based on outstanding performance and exceptional work, but the total score for the lab and the final project combined **will not exceed 20** points.

For example, if your base score is 19 and you earn 2 bonus points, your final score is still **capped at 20**.

清晰度	易于理解；使用适当的格式化，标题，和技术语言。	有组织；较小格式化或清晰度问题。	难以跟随。	理解；缺乏格式化或结构。

3. 同伴评估（2 分，由小组成员评估）

评估方法
每个小组成员将对其他成员进行评分，评分范围为 0 到 2 分：
0 分：未做出贡献或无帮助。1 分：做出了一些贡献但未达到预期。2 分：做出重大贡献且非常有帮助。每个成员将不会对自己进行评分。

个人评分计算
每个小组成员的个人评分将计算为他们的同伴给予的评分的平均值。

标准	优秀（2）	良好（1）	较差（0）
个人贡献	获得了平均从同伴那里得到的 1.5 到 2 分的评分，表示很强贡献和合作。	获得了平均分从同伴那里得到的 0.5 到 1.4 分，表明某些有贡献但伴随着明显的差距。	获得了平均从 0.5 以下的分数同行，表明最少或没有贡献。

4. 奖励说明

根据出色的表现和卓越的工作，你最多可以获得 **2 个奖励分**，用于**最终项目**，但实验室和最终项目的总成绩**将不会超过 20** 分。

例如，如果你的基础分数是 19 分，并且获得了 2 个奖励分，你的最终分数仍然**被限制在 20** 分。