CPSC 304 Project Cover Page

Milestone #: 2

Date: Mar. 3, 2025

Group Number: 114

Name	Student Number	CS Alias (userid)	Preferred Email Address
Alan Zhou	70409610	l5p5q	alanzhou318@gmail.com
Kevin Lei	60813573	l9b0l	kevinleimc@gmail.com
Wendi Liu	65854515	m5j4e	liu.wendi@yahoo.ca

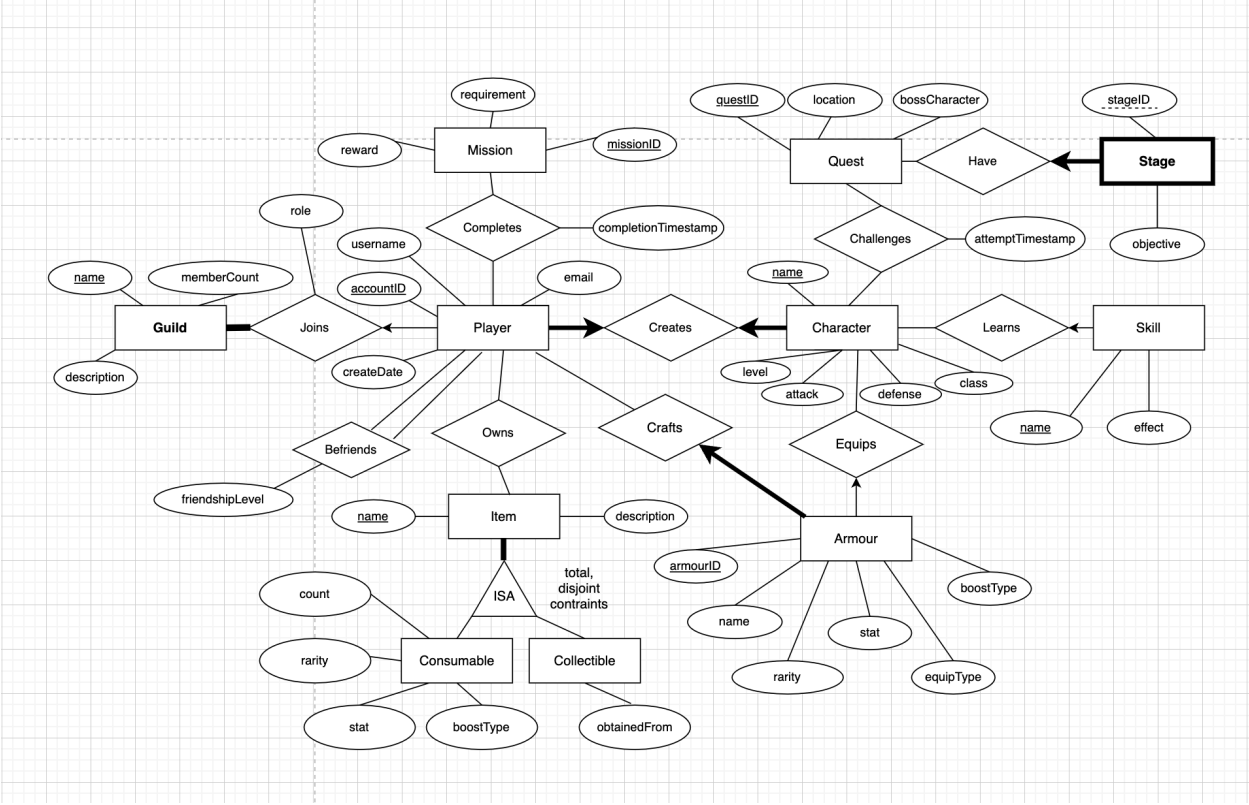
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

2. A brief (~2-3 sentences) summary of your project. Many of your TAs are managing multiple projects so this will help them remember details about your project.

Our database models the data from a generic RPG (Role-Playing Game) video game. It includes data on the player character, inventory items, mission records, and quest history divided into different stages. It also models social networks, such as friends and guilds, within the game.

3. The ER diagram you are basing your item #3 (below) on. This ER diagram may be the same as your milestone 1 submission or it might be different. If you have made changes from the version submitted in milestone 1, attach a note indicating what changes have been made and why.



- Changed the friendship relationship to be renamed as “Befriends” instead of “Has” so that it is more descriptive
- Changed the primary key of Player to only accountID, removing username and having it just be unique, but not needing both accountID and username in combination as PK
- Changed the Completes and Challenges relationships to have timestamp attributes
- Added rarity attributes to Consumable and Armour to use in non-PK/CK FDs
- Added thick line to ISA to capture total coverage constraints

4. The schema derived from your ER diagram (above). For the translation of the ER diagram to the relational model, follow the same instructions as in your lectures. The process should be reasonably straightforward. For each table:
a. List the table definition (e.g., Table1(attr1: domain1, attr2: domain2, ...)). Make sure to include the domains for each attribute.
b. Specify the primary key (PK), candidate key, (CK) foreign keys (FK), and other constraints (e.g., not null, unique, etc.) that the table must maintain.

- PlayerJoins(accountID: integer (PK), username: varchar(16) unique (CK), email: varchar, createDate: date, role: varchar, **guildName**: varchar(16) (FK referencing Guild(name)))
- Guild(name: varchar(16), memberCount: integer, description: varchar(255), **accountID**:integer (FK referencing PlayerJoins(accountID)))
  Note: total participation constraint here can't be enforced at this point
- Befriends(**account1ID**: integer (PK, FK referencing PlayerJoins(accountID)), **account2ID**: integer (PK, FK referencing PlayerJoin(accountID)), friendshipLevel: integer)
- Mission(missionID: integer (PK), requirement: varchar(255), reward: varchar(255))
- Completes(**accountID**: integer (PK, FK referencing PlayerJoins), **missionID**: integer (PK, FK referencing Mission), completionTimestamp: smalldatetime)
- Item(name: varchar(16) (PK), description: varchar(64))
- Consumable(**name**: varchar (PK, FK referencing Item), count: integer, rarity: varchar(16), stat: integer, boostType: varchar(8))
- Collectible(**name**: varchar (PK, FK referencing Item), obtainedFrom: varchar(255))
- Owns(**accountID**: integer (PK, FK referencing PlayerJoins), **name**: varchar(16) (PK, FK referencing Item))
- CreatesCharacter(name: varchar(16) (PK), level: integer, attack: integer, defence: integer, class: varchar(8), **playerID**: integer unique not null (FK referencing PlayerJoins(accountID)))
- CraftsArmour(armourID: integer (PK), name: varchar, rarity: varchar(16), stat: integer, boostType: varchar(8), equipType: varchar(8), **accountID**: integer not null (FK referencing PlayerJoins))
- Equips(**armourID**: integer (PK, FK referencing ArmourCrafts), **characterName**: varchar(16) (FK referencing Character(name))
- LearnsSkill(name: varchar (PK), effect: varchar(64), **characterName**: varchar(16) (FK referencing Character(name))
- Quest(questID: integer (PK), location: varchar, bossCharacter: varchar)
- HaveStage(stageID: integer (PK), objective: varchar, **questID**: integer (PK, FK referencing Quest))

- Challenges(**<u>characterName</u>**: varchar(16) (PK, FK referencing Character(name)), **<u>questID</u>**: integer (PK, FK referencing Quest), attemptTimestamp: smalldatetime)

5. Functional Dependencies (FDs)
a. Identify the functional dependencies in your relations, including the ones involving all candidate keys (including the primary key).
PKs and CKs are considered functional dependencies and should be included in the list of FDs. You do not need to include trivial FDs such as A -> A.
Note: In your list of FDs, there must be some kind of valid FD other than those identified by a PK or CK. If you observe that no relations have FDs other than the PK and CK(s), then you will have to intentionally add some (meaningful) attributes to show valid FDs. We want you to get a good normalization exercise.

PlayerJoins
- accountID -> username, email, createDate, role, guildName
- username -> accountID, email, createDate, role, guildName

Guild
- name -> memberCount, description

Befriends
- accountID1, accountID2 -> friendshiplevel

Mission
- missionID -> reward, requirement

Completes
- accountID, missionID -> completionTimestamp

Item
- name -> description

Consumable
- name -> boostType, rarity, count
- rarity, boostType -> stat

Collectible
- name -> obtainedFrom

Owns
- accountID -> itemName, itemDescription

CreatesCharacter
- accountID, name -> level, class, attack, defence
- level, class -> attack, defence

CraftsArmour
- armourID -> armourName, rarity, boostType, equipType, accountID

- armourName -> rarity, boostType, equipType
- rarity, boostType, equipType -> stat

Equips
- armourID -> name

LearnsSkill
- skillName -> effect, name

Quest
- questID -> location, bossCharacter
- bossCharacter -> location

HaveStage
- stageID, questID -> objective

Challenges
- characterName, questID -> attemptTimestamp

6. Normalization

a. Normalize each of your tables to be in 3NF or BCNF. Give the list of tables, their primary keys, their candidate keys, and their foreign keys after normalization. You should show the steps taken for the decomposition in a manner similar to that done in class. Should there be errors, and no work is shown, no partial credit can be awarded without steps shown.
The format should be the same as Step 3, with tables listed similar to Table1(attr1:domain1, attr2:domain2, ...). ALL Tables must be listed, not only the ones post normalization.

Highlighted tables are the final tables (ones after decomposition/normalization).

- PlayerJoins(accountID: integer (PK), username: varchar(16) unique (CK), email: varchar(255), createDate: date, role: varchar, **guildName**: varchar(16) (FK referencing Guild(name)))
  - in BCNF
- Guild(name: varchar(16), memberCount: integer, description: varchar(255))
  - in BCNF
- Befriends(**accountID1**: integer (PK, FK referencing PlayerJoins(accountID)), **account2ID**: integer (PK, FK referencing PlayerJoin(accountID)), friendshipLevel: integer)
  - In BCNF
- Mission(missionID: integer (PK), requirement: varchar(255), reward: varchar(255))
  - in BCNF
- Completes(**accountID**: integer (PK, FK referencing PlayerJoins), **missionID**: integer (PK, FK referencing Mission), completionTimestamp: date)

    - in BCNF
- Item(<u>name</u>: varchar(16) (PK), description: varchar(64))
    - in BCNF
- Consumable(**<u>name</u>**: varchar (PK, FK referencing Item), count: integer, rarity: varchar(16), stat: integer, boostType: varchar(8))
    - rarity, boostType -> stat *<u>violates BCNF</u>*
        - ConsumableStat(<u>rarity</u>: varchar(16) (PK), stat: integer, <u>boostType</u>: varchar(8) (PK))
            - in BCNF
        - Consumable(**<u>name</u>**: varchar (PK, FK referencing Item), count: integer, rarity: varchar(16), boostType: varchar(8))
            - in BCNF
- Collectible(**<u>name</u>**: varchar (PK, FK referencing Item), obtainedFrom: varchar(255))
    - in BCNF
- Owns(**<u>accountID</u>**: integer (PK, FK referencing PlayerJoins), **<u>itemName</u>**: varchar(16) (PK, FK referencing Item))
    - In BCNF
- CreatesCharacter(<u>name</u>: varchar(16) (PK), level: integer, attack: integer, defence: integer, class: varchar(8), **playerID**: integer unique not null (FK referencing PlayerJoins(accountID)))
    - level, class -> attack, defence *<u>violates BCNF</u>*
        - CharacterStats(<u>level</u>: integer (PK), attack: integer, defence: integer, <u>class</u>: varchar(8) (PK))
            - in BCNF
        - CreatesCharacter(<u>name</u>: varchar(16) (PK), level: integer, class: varchar(8), **playerID**: integer unique not null (FK referencing PlayerJoins(accountID))
            - in BCNF

- CraftsArmour(armourID: integer (PK), name: varchar, rarity: varchar(16), stat: integer, boostType: varchar(8), equipType: varchar(8), **accountID**: integer not null (FK referencing PlayerJoins))
    - rarity, boostType, equipType -> stat *violates BCNF*
        - ArmourStat(rarity: varchar(16) (PK), stat: integer, boostType: varchar(8) (PK), equipType: varchar(8) (PK))
        - CraftsArmour(armourID: integer (PK), name: varchar, rarity: varchar(16), boostType: varchar(8), equipType: varchar(8), **accountID**: integer not null (FK referencing PlayerJoins))
            - armourName -> rarity, boostType, equipType violates BCNF
                - ArmourName(name: varchar(16) (PK), rarity: varchar(16), equipType: varchar(8), boostType: varchar(8)
                    - in BCNF
                - CraftsArmour(armourID: integer (PK), name: varchar(16), boostType: varchar(8), **accountID**: integer not null (FK referencing PlayerJoins))
                    - in BCNF

- Equips(**armourID**: integer (PK, FK referencing ArmourCrafts), **characterName**: varchar(16) (FK referencing Character(name))
    - in BCNF
- LearnsSkill(name: varchar (PK), effect: varchar(64), **characterName**: varchar(16) (FK referencing Character(name))
    - in BCNF

- Quest(questID: integer (PK), location: varchar, bossCharacter: varchar)
    - bossCharacter -> location violates BCNF
        - QuestLocation(location: varchar, bossCharacter: varchar(255) (PK))
            - in BCNF
        - QuestInfo(questID: integer (PK), bossCharacter: varchar(255))
            - in BCNF
- HaveStage(stageID: integer (PK), objective: varchar, **questID**: integer (PK, FK referencing Quest))
    - in BCNF
- Challenges(**characterName**: varchar(16) (PK, FK referencing Character(name)), **questID**: integer (PK, FK referencing Quest), attemptTimestamp: smalldatetime)
    - in BCNF

7. The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc.
Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use VARCHAR as the number of characters in a course name can vary greatly.

```
CREATE TABLE Guild (
name VARCHAR(16) PRIMARY KEY,
memberCount INTEGER,
description VARCHAR(255)
);

CREATE TABLE PlayerJoins (
accountID INTEGER PRIMARY KEY,
username VARCHAR(16) UNIQUE,
email VARCHAR(255),
createDate DATE,
role VARCHAR(255),
guildName VARCHAR(16),
FOREIGN KEY (guildName) REFERENCES Guild(name)
);


CREATE TABLE Befriends (
        account1ID INTEGER,
        account2ID INTEGER,
        friendshipLevel INTEGER,
        PRIMARY KEY (account1ID, account2ID),
FOREIGN KEY (account1ID) REFERENCES PlayerJoins(accountID),
FOREIGN KEY (account2ID) REFERENCES PlayerJoins(accountID)
);
```

```
CREATE TABLE Mission (
missionID INTEGER PRIMARY KEY,
requirement VARCHAR(255),
reward VARCHAR(255)
);

CREATE TABLE Completes (
accountID INTEGER,
missionID INTEGER,
completionTimestamp DATE,
PRIMARY KEY (accountID, missionID),
FOREIGN KEY (accountID) REFERENCES PlayerJoins(accountID),
FOREIGN KEY (missionID) REFERENCES Mission(missionID)
);

CREATE TABLE Item (
name VARCHAR(30) PRIMARY KEY,
description VARCHAR(64)
);

CREATE TABLE ConsumableStat (
rarity VARCHAR(16),
stat INTEGER,
boostType VARCHAR(8),
PRIMARY KEY (rarity, boostType)
);

CREATE TABLE Consumable (
name VARCHAR(16),
count INTEGER,
rarity VARCHAR(16),
boostType VARCHAR(8),
PRIMARY KEY (name),
FOREIGN KEY (name) REFERENCES Item(name)
);
```

```
CREATE TABLE Collectible (
name VARCHAR(16),
obtainedFrom VARCHAR(255),
PRIMARY KEY (name),
FOREIGN KEY (name) REFERENCES Item(name)
);

CREATE TABLE Owns (
accountID INTEGER,
name VARCHAR(16),
PRIMARY KEY (accountID, name),
FOREIGN KEY (accountID) REFERENCES PlayerJoins(accountID),
FOREIGN KEY (name) REFERENCES Item(name)
);

CREATE TABLE CharacterStats (
level INTEGER,
attack INTEGER,
defence INTEGER,
class VARCHAR(8),
PRIMARY KEY (level, class)
);

CREATE TABLE CreatesCharacter (
name VARCHAR(16) PRIMARY KEY,
level INTEGER,
class VARCHAR(10),
playerID INTEGER UNIQUE NOT NULL,
FOREIGN KEY (playerID) REFERENCES PlayerJoins(accountID)
);

CREATE TABLE ArmourName (
name VARCHAR(16) PRIMARY KEY,
rarity VARCHAR(16),
equipType VARCHAR(8),
boostType VARCHAR(8)
);
```

```
CREATE TABLE ArmourStat (
rarity VARCHAR(16),
stat INTEGER,
boostType VARCHAR(8),
equipType VARCHAR(8),
PRIMARY KEY (rarity, boostType, equipType)
);

CREATE TABLE CraftsArmour (
armourID INTEGER PRIMARY KEY,
name VARCHAR(16),
boostType VARCHAR(8),
accountID INTEGER NOT NULL,
FOREIGN KEY (accountID) REFERENCES PlayerJoins(accountID)
);

CREATE TABLE Equips (
armourID INTEGER PRIMARY KEY,
characterName VARCHAR(16),
FOREIGN KEY (armourID) REFERENCES CraftsArmour(armourID),
FOREIGN KEY (characterName) REFERENCES CreatesCharacter(name)
);

CREATE TABLE LearnsSkill (
name VARCHAR(255) PRIMARY KEY,
effect VARCHAR(64),
characterName VARCHAR(16),
FOREIGN KEY (characterName) REFERENCES CreatesCharacter(name)
);

CREATE TABLE QuestLocation (
location VARCHAR(255),
bossCharacter VARCHAR(255) PRIMARY KEY
);
```

```
CREATE TABLE QuestInfo (
questID INTEGER PRIMARY KEY,
bossCharacter VARCHAR(255)
);

CREATE TABLE HaveStage (
stageID INTEGER PRIMARY KEY,
objective VARCHAR(255),
questID INTEGER,
FOREIGN KEY (questID) REFERENCES QuestInfo(questID)
);

CREATE TABLE Challenges (
characterName VARCHAR(16),
questID INTEGER,
attemptTimestamp DATE,
PRIMARY KEY (characterName, questID),
FOREIGN KEY (characterName) REFERENCES CreatesCharacter(name),
FOREIGN KEY (questID) REFERENCES QuestInfo(questID)
);
```

8. INSERT statements to populate each table with at least 5 tuples. You will likely want to have more than 5 tuples so that you can have meaningful queries later.

```
INSERT INTO Guild
    VALUES
    ('Goats Guild', 3, 'A guild for goats all around the world'),
    ('Best Players', 2, 'Only the best players are in this guild'),
    ('Guild 1', 1, 'guild 1 description'),
    ('Guild 2', 1, 'guild 2 description'),
    ('Guild 3', 1, 'guild 3 description');


INSERT INTO PlayerJoins
    VALUES
    (1, 'playerone', 'iplaygames@email.com', '2025-01-01', 'member', 'Best Players'),
```

(2, 'thegoat', 'goatedgamer@email.com', '2025-01-05', 'leader', 'Goats Guild'),
(3, 'legoat', 'goatedgamer@email.com', '2025-02-05', 'deputy', 'Goats Guild'),
(4, 'dabest', 'bestest@email.com', '2025-02-20', 'leader', 'Best Players'),
(5, 'mountaingoat', 'mountaingoat@email.com', '2025-02-21', 'member', 'Goats Guild'),
(6, 'solo', 'soloplayer@email.com', '2025-02-22', NULL, NULL),
(7, 'lesunshine', 'a@gmail.com', '2025-02-22', 'leader', 'Guild 1'),
(8, 'lebron', 'b@gmail.com', '2025-02-22', 'leader', 'Guild 2'),
(9, 'leGM', 'c@gmail.com', '2025-02-22', 'leader', 'Guild 3');

INSERT INTO Befriends
VALUES
(1, 2, 10),
(2, 3, 23),
(2, 5, 5),
(3, 5, 1),
(7, 8, 5);

INSERT INTO Mission
VALUES
(101, 'kill 3 monsters', '10 coins'),
(102, 'defeat a boss character', 'crown'),
(103, 'add a friend', 'friendship bracelet'),
(104, 'finish all quests', '100000 coins'),
(105, 'finish first quest', '10 coins');

INSERT INTO Completes
VALUES
(1, 101, '2025-01-01 15:30:10'),
(1, 102, '2025-01-01 15:35:33'),
(1, 103, '2025-01-05 22:00:05'),
(2, 104, '2025-01-01 13:35:33'),
(3, 104, '2025-01-05 20:00:05');

INSERT INTO Item

VALUES

('Bracelet', 'A token of your friendship'),
('Crown', 'Reward for defeating your first boss'),
('Ancient Coin', 'A rare coin from ancient times'),
('Golden Feather', 'A shimmering feather from a rare bird'),
('Dragon Scale', 'A tough scale from a defeated dragon'),

('Sandwich', 'A tasty sandwich that boosts attack'),
('Strength potion', 'A potion that increases your strength temporarily'),
('Healing Herb', 'A magical herb that restores health'),
('Energy Drink', 'A beverage that restores stamina'),
('Elixir of Wisdom', 'A rare potion that increases mana');

INSERT INTO ConsumableStat

VALUES

('common', 2, 'attack'),
('common', 4, 'defence'),
('rare', 5, 'attack'),
('rare', 8, 'defence'),
('legendary', 10, 'attack');

INSERT INTO Consumable

VALUES

('Sandwich', 5, 'common', 'attack'),
('Strength potion', 2, 'rare', 'attack'),
('Healing Herb', 3, 'uncommon', 'health'),
('Energy Drink', 4, 'common', 'stamina'),
('Elixir of Wisdom', 1, 'legendary', 'mana');

INSERT INTO Collectible

VALUES

('Bracelet', 'Mission reward'),
('Crown', 'Mission reward'),
('Ancient Coin', 'Hidden treasure'),

('Golden Feather', 'Rare bird drop'),
('Dragon Scale', 'Defeated a dragon');

INSERT INTO Owns
VALUES
(1, 'Bracelet'),
(1, 'Crown'),
(2, 'Bracelet'),
(3, 'Bracelet'),
(5, 'Bracelet'),
(7, 'Bracelet'),
(8, 'Bracelet'),
(4, 'Ancient Coin'),
(6, 'Dragon Scale');

INSERT INTO CharacterStats
VALUES
(1, 5, 3, 'fighter'),
(2, 6, 4, 'fighter'),
(3, 8, 4, 'fighter'),
(1, 3, 5, 'tank'),
(2, 4, 6, 'tank'),
(3, 4, 8, 'tank'),
(1, 7, 1, 'assassin'),
(2, 8, 2, 'assassin'),
(3, 9, 3, 'assassin');

INSERT INTO CreatesCharacter
VALUES
('Bob', 3, 'fighter', 1),
('Joe', 3, 'tank', 2),
('Alice', 2, 'mage', 3),
('Eve', 5, 'rogue', 4),
('Max', 1, 'archer', 5),
('Luna', 4, 'healer', 6),
('Kai', 6, 'summoner', 7),

('Zane', 2, 'berserker', 8),
('Cody', 2, 'berserker', 9);

INSERT INTO ArmourName
VALUES
('Light headgear', 'common', 'helmet', 'defence'),
('Nice helmet', 'rare', 'helmet', 'defence'),
('Breastplate', 'rare', 'upper', 'attack'),
('Fleet footwear', 'rare', 'boots', 'attack'),
('Cool Shoes', 'epic', 'boots', 'attack');

INSERT INTO CraftsArmour
VALUES
(1, 'Light headgear', 'defence', 1),
(2, 'Nice helmet', 'defence', 1),
(3, 'Breastplate', 'attack', 1),
(4, 'Breastplate', 'defence', 2),
(5, 'Fleet footwear', 'attack', 5);

INSERT INTO Equips
VALUES
(2, 'Bob'),
(3, 'Bob'),
(4, 'Joe'),
(5, 'Joe'),
(1, 'Joe');

9. An explicit acknowledgment about your use of AI tools in this assignment. Specifically, we are looking for a clear yes/no about whether you have used one or more AI tools. If yes, we want to know which tool(s) you have used and what prompt(s) you have given the tool.

We have not used AI tools in this assignment.