**University of British Columbia, Vancouver**
Department of Computer Science

# CPSC 304 Project Cover Page

Milestone #: 2

Date: 1th March 2024

Group Number: 10

| Name | Student Number | CS Alias (Userid) | Preferred E-mail Address |
|---|---|---|---|
| Kanish Khanna | 20186961 | e2p2p@ugrad.cs.ubc.ca | Kanishkhanna2020@gmail.com |
| Yiquan Liu | 33205998 | c7m8c@ugrad.cs.ubc.ca | Springliu2003@gmail.com |
| Aaditya Suri | 41935511 | f5o3r@ugrad.cs.ubc.ca | Adityasuri01@gmail.com |

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

**University of British Columbia, Vancouver**
Department of Computer Science

---

1. **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.**

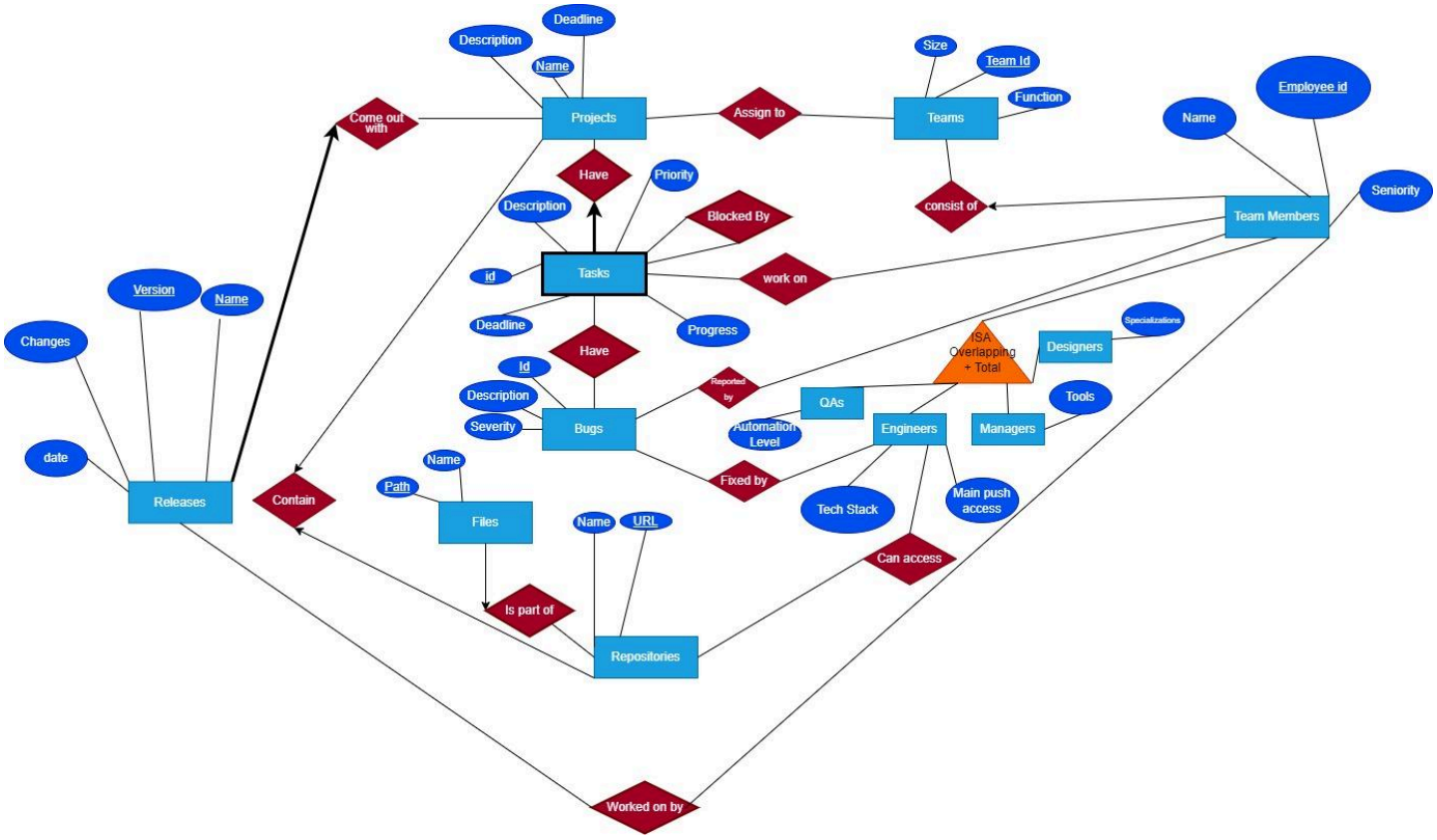
   The domain for our project is Software Engineering project management and tracking. We want to mainly focus on tracking the tasks, blockers, bugs, repositories, deadlines and versions for software engineering teams. Although we have a component related to tracking employees and departments, we do not intend to make it the main emphasis of our application.

   The aim of the application is to aggregate and store all this data in a single place and be able to query it efficiently. This can resolve situations where two disjoint teams are working on the same project and need to collaborate efficiently to distribute tasks and solve bugs or view changes in various software releases over the years and be able to contact the people who worked on a previous release.

**2. 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. If you have decided not to implement the suggestions given by your project mentor, please be sure to leave a note stating why. This is not to say that you must do everything that your project mentor says. In many instances, there are trade-offs between design choices and your decision may be influenced by different factors. Your TAs will often leave suggestions that are meant to help massage your project into a form that will fit with the requirements in future project milestones. If you choose not to take their advice, it would be helpful for them to know why to better assist the group moving forward.**

**ER Diagram:**

**University of British Columbia, Vancouver**
Department of Computer Science




Changes made to the ER Diagram:

Since we were missing meaningful attributes for the ISA subclasses, we added Automation Level for QAs, Tools for Managers, Specializations for Designers. ISA constraints are overlapping and total. TaskID is underlined with dotted line.

**3. 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, ...)).p 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.**

a) List of Table definitions:
*Primary Key* is represented as <u>underlined</u>

---

*Foreign Key* is represented as ***bold***

Projects (Name: varchar, Description: varchar, Deadline: date)

Teams (TeamID: integer, Size: integer, Function: varchar)

AssignTo (**Name: varchar, TeamID: integer**)

TeamMembers (EmployeeID: integer, **TeamID: integer**, Name: varchar, Seniority: integer)

WorkOnBy (**EmployeeID: integer, Version: integer, ReleaseName: varchar**)

Releases (Version:varchar, Name:varchar, **ProjectName: varchar**, Changes: varchar, Date: date)

Repositories (URL: varchar, **ProjectName: varchar**, Name:varchar)

Files(Path:varchar, **URL:varchar**, FileName:varchar)

ReportedBy(**BugID:integer, EmployeeID:integer**)

Bugs(ID: integer, Description: varchar, Severity: integer)

TaskHaveBugs(**BugID: integer, TaskID: integer, ProjectName: varchar**)

Tasks(ID: integer, **ProjectName: varchar**, Deadline: date, Progress: integer, Description: varchar, Priority: integer)

BlockedBy(**BlockerTaskID: integer, BlockedTaskID: integer**)

Engineers(**EmployeeID: integer**, TechStack: varchar, MainPushAccess: boolean)

QAs(**EmployeeID: integer**, AutomationLevel: integer)

Managers(**EmployeeID: integer,** Tools: varchar)

Designers(**EmployeeID: integer**, Specialization: varchar)

b)

| | **PRIMARY KEY** | **CANDIDATE KEY** | **FOREIGN KEY** | **SPECIAL ATTRIBUTES** |
|---|---|---|---|---|
| Projects | Name | Name | | Description UNIQUE |
| Teams | Team ID | Team ID | | |
| AssignTo | Name, Team ID | Name, Team ID | Name, Team ID | |
| TeamMembers | EmployeeID | EmployeeID | TeamID | |
| WorkOnBy | EmployeeID, Version, Name | EmployeeID, Version, Name | EmployeeID, Version, ReleaseName | |
| Releases | Version, Name | Version, Name | ProjectName | |
| Repositories | URL | URL | ProjectName | |
| Files | Path | Path | URL | URL NOT NULL |
| ReportedBy | BugID, EmployeeID | BugID, EmployeeID | BugID, EmployeeID | |
| Bugs | ID | ID | | |
| TaskHaveBugs | BugID, TaskID, ProjectName | BugID, TaskID, ProjectName | BugID, TaskID, ProjectName | |
| Tasks | ID, ProjectName | ID, ProjectName | ProjectName | |
| BlockedBy | BlockerTaskID, BlockedTaskID | BlockerTaskID, BlockedTaskID | BlockerTaskID, BlockedTaskID | |
| Engineers | EmployeeID | EmployeeID | EmployeeID | EmployeeID NOT NULL |
| QA's | EmployeeID | EmployeeID | EmployeeID | EmployeeID NOT NULL |
| Managers | EmployeeID | EmployeeID | EmployeeID | EmployeeID NOT NULL |

| | | | | |
|---|---|---|---|---|
| Designers | EmployeeID | EmployeeID | EmployeeID | EmployeeID NOT NULL |

**4. 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 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. Your design must go through a normalization process. You do not need to have a non-PK/CK FD for each relation but be reasonable. If your TA feels that some non-PK/CK FDs have been omitted, your grade will be adjusted accordingly.**

FUNCTIONAL DEPENDENCIES FOR EACH RELATION

Project:

ProjectName -> Description

ProjectName -> Deadline

Team:

TeamID -> Size

TeamID -> Function

Team Member:

EmployeeID -> Name

EmployeeID -> Seniority

EmployeeID -> TeamID

Release:

ReleaseVersion, ReleaseName -> ProjectName

ReleaseVersion, ReleaseName -> Changes

ReleaseVersion, ReleaseName -> Date

Respository:

RespoURL -> ProjectName

RespoURL -> RespoName

File:

FilePath -> FileName

FilePath -> URL

Bug:

BugID -> Description

BugID -> Severity

Task:

TaskID, ProjectName -> Deadline

TaskID, ProjectName -> Progress

TaskID, ProjectName -> Description

TaskID, ProjectName -> Priority

Priority -> Deadline

Description -> Priority

**5. 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. 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.**

Decompose the above into BCNF
R(TaskID, ProjectName, Deadline, Progress, Description, Priority)

(FD1)TaskID, ProjectName -> Deadline
(FD2)TaskID, ProjectName -> Progress
(FD3)TaskID, ProjectName -> Description
(FD4)TaskID, ProjectName -> Priority
(FD5)Priority -> Deadline
(FB6)Description -> Priority

TaskID, ProjectName$^+$ ={TaskID, ProjectName, Deadline, Progress, Description, Priority}
Priority$^+$ = {Priority, Deadline}
Description$^+$ = {Description, Priority, Deadline}

Since the closure of TaskID and ProjectName together include all attributes of R, FD1-FD4 are in BCNF. FD5 and FD6 violate BCNF because Priority and Description are not superkeys.

1. Decompose FD5 into R1(<u>Priority</u>, Deadline)  R2(TaskID, ProjectName, Progress, Description, Priority)
   R1 is in BCNF with Priority as the superkey, R2 is not because description is not a superkey
2. Decompose FD6 into R3(<u>Description</u>, Priority)  R4(Description, <u>TaskID, ProjectName</u>, Progress)
   R3 is in BCNF with Description as the superkey, R4 is also in BCNF as no FDs in R4 violates BCNF, with TaskID + ProjectName as the superkey

   Final Relations: R1(<u>Priority</u>, Deadline), R3(<u>Description</u>, Priority), R4(Description, <u>TaskID, ProjectName</u>, Progress)

Therefore, after Normalization, all the relations will be:

---

**None of these relations needed normalization as they were already in BCNF**

Projects (Name: varchar, Description: varchar, Deadline: date)

Teams (TeamID: integer, Size: integer, Function: varchar)

AssignTo (**Name: varchar, TeamID: integer**)

TeamMembers (Employee ID: integer, **TeamID: integer**, Name: varchar, Seniority: integer)

WorkOnBy (**EmployeeID: integer, Version: integer, ReleaseName: varchar**)

Releases (Version:varchar, Name:varchar, **ProjectName: varchar**, Changes: varchar, Date: date)

Repositories (URL: varchar, **ProjectName: varchar**, Name:varchar)

Files(Path:varchar, **URL:varchar**, FileName:varchar)

ReportedBy(**BugID:integer, EmployeeID:integer**)

Bugs(ID: integer, Description: varchar, Severity: integer)

TaskHaveBugs(**BugID: integer, TaskID: integer, ProjectName: varchar**)

BlockedBy(**BlockerTaskID: integer, BlockedTaskID: integer**)

Engineers(**EmployeeID: integer**, TechStack: varchar, MainPushAccess: boolean)

QAs(**EmployeeID: integer**, AutomationLevel: integer)

Managers(**EmployeeID: integer,** Tools: varchar)

Designers(**EmployeeID: integer**, Specialization: varchar)

**Only the Tasks relation was normalized**

PostNormTasksR1(**Priority: integer**, Deadline: date)

PostNormTasksR2(**Description: varchar**, Priority: integer)

PostNormTasksR3(TaskID: integer, ProjectName: varchar, Description: varchar, Progress: integer)

**6. 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.**

**Note**: **Oracle doesn't support 'ON UPDATE CASCADE' commands, but we are including it as convention.**

```
CREATE TABLE Projects (
        Name VARCHAR,
        Description VARCHAR UNIQUE,
        Deadline DATE,
        PRIMARY KEY (Name)
)
CREATE TABLE Teams (
        TeamID  INTEGER,
        Size INTEGER,
        Function VARCHAR,
        PRIMARY KEY (TeamID)
)

CREATE TABLE AssignTo (
        Name VARCHAR,
        TeamID INTEGER,
        PRIMARY KEY (Name, TeamID),
        FOREIGN KEY (Name)
                REFERENCES Projects(Name),
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
        FOREIGN KEY (TeamID)
```

```
                REFERENCES Teams(TeamID)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
)


CREATE TABLE TeamMembers (
        EmployeeID INTEGER,
        TeamID      INTEGER,
        Name        VARCHAR,
        Seniority   INTEGER,
        PRIMARY KEY (EmployeeID)
        FOREIGN KEY (TeamID)
                REFERENCES Teams (TeamID)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
);

CREATE TABLE WorkOnBy (
        EmployeeID          INTEGER,
        Version             INTEGER,
        ReleaseName         VARCHAR,
        PRIMARY KEY (EmployeeID, Version, ReleaseName),
        FOREIGN KEY (EmployeeID)
                REFERENCES TeamMembers(EmployeeID)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
        FOREIGN KEY(Version)
                REFERENCES Releases(Version)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
        FOREIGN KEY(ReleaseName)
                REFERENCES Releases(Name)
                        ON DELETE CASCADE
```

```
                    ON UPDATE CASCADE
);

CREATE TABLE Releases (
        Version      VARCHAR,
        Name         VARCHAR,
        ProjectName VARCHAR  NOT NULL,
        Changes      VARCHAR,
        Date         DATE,
        PRIMARY KEY (Version)
        FOREIGN KEY (ProjectName)
            REFERENCES Projects(Name)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);



CREATE TABLE Repositories (
  URL          VARCHAR PRIMARY KEY,
  ProjectName VARCHAR NOT NULL,
  Name         VARCHAR,
  FOREIGN KEY (ProjectName)
        REFERENCES Projects(Name)
            ON DELETE CASCADE
            ON UPDATE CASCADE
);

CREATE TABLE Files (
  Path       VARCHAR,
  URL        VARCHAR NOT NULL,
  FileName VARCHAR,
  PRIMARY KEY (Path),
  FOREIGN KEY (URL)
```

```
        REFERENCES Repositories(URL)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE ReportedBy (
  BugID         INTEGER,
  EmployeeID INTEGER NOT NULL,
  PRIMARY KEY (BugID, EmployeeID),
  FOREIGN KEY (BugID)
        REFERENCES Bugs(ID),
                ON DELETE CASCADE
                ON UPDATE CASCADE
  FOREIGN KEY (EmployeeID)
        REFERENCES Employees(EmployeeID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Bugs (
  ID              INTEGER,
  Description VARCHAR,
  Severity      INTEGER,
  PRIMARY KEY (ID)
);

CREATE TABLE TaskHaveBugs (
  BugID INTEGER,
  TaskID INTEGER,
  ProjectName VARCHAR,
  PRIMARY KEY (BugID, TaskID, ProjectName),
  FOREIGN KEY(BugID)
        REFERENCES Bugs(ID),
                ON DELETE CASCADE
```

```
                ON UPDATE CASCADE
  FOREIGN KEY(TaskID)
        REFERENCES Tasks(ID),
                ON DELETE CASCADE
                ON UPDATE CASCADE
  FOREIGN KEY(ProjectName)
        REFERENCES Projects(Name)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE PostNormTasksR1 (
        Priority  INTEGER,
        Deadline DATE,
        PRIMARY KEY(Priority)
        FOREIGN KEY(Priority)
              REFERENCES PostNormTaskR2
                    ON DELETE CASCADE
                    ON UPDATE CASCADE
);

CREATE TABLE PostNormTasksR2 (
        Description VARCHAR,
        Priority  INTEGER,
        PRIMARY KEY(Description)
        FOREIGN KEY(Description)
                REFERENCES PostNormTasksR3(Description)
                     ON DELETE CASCADE
                     ON UPDATE CASCADE
);
CREATE TABLE PostNormTasksR3 (
        Description      VARCHAR,
        TaskID           INTEGER,
        ProjectName    VARCHAR,
```

```
        Progress        INTEGER,
        PRIMARY KEY (TaskID, ProjectName)
);



CREATE TABLE BlockedBy (
  BlockerTaskID INTEGER,
  BlockedTaskID INTEGER,
  PRIMARY KEY (BlockerTaskID, BlockedTaskID),
  FOREIGN KEY(BlockerTaskID)
        REFERENCES Tasks(ID),
                ON DELETE CASCADE
                ON UPDATE CASCADE
  FOREIGN KEY(BlockedTaskID)
        REFERENCES Tasks(ID)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);

CREATE TABLE Engineers (
  EmployeeID INTEGER NOT NULL,
  TechStack VARCHAR,,
  MainPushAccess BOOLEAN,
  PRIMARY KEY (EmployeeID),
  FOREIGN KEY(EmployeeID)
        REFERENCES TeamMembers
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
CREATE TABLE QAs (
  EmployeeID INTEGER NOT NULL,
  AutomationLevel INTEGER,
  PRIMARY KEY (EmployeeID),
```

```
  FOREIGN KEY(EmployeeID)
        REFERENCES TeamMembers
              ON DELETE CASCADE
              ON UPDATE CASCADE
);

CREATE TABLE Managers (
  EmployeeID INTEGER NOT NULL,
  Tools VARCHAR,
  PRIMARY KEY (EmployeeID),
  FOREIGN KEY(EmployeeID)
        REFERENCES TeamMembers
              ON DELETE CASCADE
              ON UPDATE CASCADE
);

CREATE TABLE Designers (
  EmployeeID INTEGER NOT NULL,
  Specialization VARCHAR,
  PRIMARY KEY (EmployeeID),
  FOREIGN KEY (EmployeeID)
        REFERENCES TeamMembers
              ON DELETE CASCADE
              ON UPDATE CASCADE
);
```

**7. 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 Projects (Name, Description, Deadline)
VALUES ('Search','Search Engine', '2024-03-15'),
          ('Ads', Ad integration', '2024-04-30'),
```

```
        ('Networking', 'Network routing', '2024-05-20'),
        ('Classifier', 'CNN image classification', '2024-06-10'),
        ('Frontend', 'UI for website', '2024-07-25');


INSERT INTO Teams (TeamID, Size, Function)
VALUES (1, 5, 'Development'),
        (2, 8, 'Testing'),
        (3, 6, 'Design'),
        (4, 7, 'Support'),
        (5, 4, 'Marketing');

INSERT INTO AssignTo (Name, TeamID)
VALUES ('Search', 1),
        ('Ads', 2),
        ('Networking', 3),
        ('Classifier', 4),
        ('Frontend', 5);

INSERT INTO TeamMembers (EmployeeID, TeamID, Name, Seniority)
VALUES (1, 1, 'John', 3),
        (2, 1, 'Alice', 2),
        (3, 3, 'Bob', 4),
        (4, 4, 'Emma', 1),
        (5, 4, 'Tom', 5);


INSERT INTO WorkOnBy (EmployeeID, Version, VersionName)
VALUES (1, 1, 'Release1'),
        (2, 2, 'Release2'),
        (3, 3, 'Release3'),
        (4, 3, 'Release3'),
        (5, 3, 'Release3');

INSERT INTO Releases (Version, Name, ProjectName, Changes, Date)
```

```
VALUES (1, 'Release1', 'Search', 'Bug fixes and enhancements', '2024-03-10'),
       (2, 'Release2', 'Ads', 'New feature additions', '2024-04-20'),
       (3, 'Release3', 'Networking', 'Performance improvements', '2024-05-15'),
       (4, 'Release4', 'Classifier', 'Major overhaul and redesign', '2024-06-30'),
       (5, 'Release5', 'Frontend', 'Marketing campaign updates', '2024-07-20');

INSERT INTO Repositories (URL, ProjectName, Name)
VALUES ('http://example.com/repo1', 'Search', 'Repo1'),
       ('http://example.com/repo2', 'Ads, 'Repo2'),
       ('http://example.com/repo3', 'Networking', 'Repo3'),
       ('http://example.com/repo4', 'Classifier, 'Repo4'),
       ('http://example.com/repo5', Frontend, 'Repo5');

INSERT INTO Files (Path, URL, FileName)
VALUES ('/path1', 'http://example.com/repo1', 'file1.txt'),
       ('/path2', 'http://example.com/repo2', 'file2.txt'),
       ('/path3', 'http://example.com/repo3', 'file3.txt'),
       ('/path4', 'http://example.com/repo4', 'file4.txt'),
       ('/path5', 'http://example.com/repo5', 'file5.txt');

INSERT INTO ReportedBy (BugID, EmployeeID)
VALUES (1, 1),
       (2, 2),
       (3, 3),
       (4, 4),
       (5, 5);

INSERT INTO Bugs (ID, Description, Severity)
VALUES (1, 'Bug description 1', 3),
       (2, 'Bug description 2', 2),
       (3, 'Bug description 3', 1),
       (4, 'Bug description 4', 2),
       (5, 'Bug description 5', 3);

INSERT INTO TaskHaveBugs (BugID, TaskID, ProjectName)
```

```
VALUES (1, 1, 'Search'),
       (2, 2, 'Ads'),
       (3, 3, 'Networking'),
       (4, 4, 'Classifier'),
       (5, 5, 'Frontend');


INSERT INTO PostNormTasksR1 (Priority, Deadline)
VALUES (1, '2024-03-20'),
       (2, '2024-04-25'),
       (3, '2024-05-30'),
       (4, '2024-06-15'),
       (5, '2024-07-30');

INSERT INTO PostNormTasksR2 (Description, Priority)
VALUES (‘Task description 1’, 1),
       (‘Task description 2’, 2),
       (‘Task description 3’, 3),
       (‘Task description 4’, 4),
       (‘Task description 4’, 5);

INSERT INTO PostNormTasksR3 (TaskID, ProjectName, Description, Progress)
VALUES (1, 'Search', 'Task description 1', 50),
       (2, 'Ads', 'Task description 2', 75),
       (3, 'Networking', 'Task description 3', 30),
       (4, 'Classifier', 'Task description 4', 90),
       (5, 'Frontend', 'Task description 5', 60);


INSERT INTO BlockedBy (BlockerTaskID, BlockedTaskID)
VALUES (1, 2),
       (2, 3),
       (3, 4),
       (4, 5),
```

```
        (1, 5);

INSERT INTO Engineers (EmployeeID, TechStack, MainPushAccess)
VALUES (1, 'Java, Spring', true),
        (2, 'Python, Django', false),
        (3, 'JavaScript, React', true),
        (4, 'C#, .NET', false),
        (5, 'Ruby, Rails', true);

INSERT INTO QAs (EmployeeID, AutomationLevel)
VALUES (1, 3),
        (2, 2),
        (3, 1),
        (4, 2),
        (5, 3);

INSERT INTO Managers (EmployeeID, Tools)
VALUES (1, 'Jira, Confluence'),
        (2, 'Trello, Slack'),
        (3, 'Asana, Basecamp'),
        (4, 'GitHub, GitLab'),
        (5, 'Bitbucket, Jenkins');

INSERT INTO Designers (EmployeeID, Specialization)
VALUES (1, 'UI/UX design'),
        (2, 'Graphic design'),
        (3, 'Web design'),
        (4, 'Product design'),
        (5, 'Interior design');
```