**Database Design-Package Diagram**

**Version 2.0**

**Project Management App**

**Team A**

**CSC-355**

**Spring 2016**



04/08/2016

Author: Tyler Mariano – System Designer & Jennifer Li – Project Leader

Submitted To: Dr. Joo Tan

**TABLE OF CONTENTS**

Revision History…………………………………………………………………………………..ii

1.0 Introduction……..….....…………………………….…...…………………………….……....1

2.0 Database System………………………………………………………………………………1

2.1 Overview………………………………………………………………………………1

2.2 Type of Database……………………………………………………………………...1

2.3 Tables and Attributes………………………………………………………………….1

2.4 ER Diagram…………………………………………………………………………...2

2.5 Table Details…………………………………………………………………………..3

2.5.1 UserTable…………………………………………………………………...3

2.5.2 ProjrctTable…………………………………………………………………4

2.5.3 TaskTable…………………………………………………………………...5

2.5.4 ProgressTable……………………………………………………………….6

**REVISION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Description | Date |
| 1.0 | Jennifer Li | I created the first draft. | 1/31/2016 |
|  | Jennifer Li | Created tables outlines | 02/15/2016 |
|  | Jennifer Li | Added Section 1 and 2.2 | 02/16/2016 |
| 1.1 | Tyler Mariano | I added descriptions to each table and completed the overview. Also, added attribute descriptions and examples. | 02/17/2016 |
| 1.2 | Jennifer Li | Fixed some of the differences in the tables. EX- changed the names from Progress to TaskProgress.  Added ProjectProgress and TaskDependancy | 02/17/2016 |
| 2.0 | Tyler Mariano | I Added the suggested edits given to our group by Dr. Tan. | 04/01/2016 |

**1.0 INTRODUCTION**

The purpose of this document is to explain the design of the Project Management App database, show the relationships between each of the database entities, and illustrate how they interact with the application. This document also highlights specific details regarding the implementation of the software and technologies that will be used in this project.

**2.0 Database System**

A database can be defined/described in many ways. The diagrams to follow contain a basic overview of what database software, tables, and relationships are being using to create the Project Management database. There is an entity–relationship, or ER-Diagram, to visually show table relationships. There is also table descriptions to help explain their functionality and purposes. Lastly, each of the database table’s attributes have explanations and examples.

**2.1 Overview**

The Project Management database will be implemented using SQLite. SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The database will be created using the sqlitebrowser software. This software will be used for storage and maintenance of the database. Using this software will allow for easier database manipulation in the future. Also, SQLite is compatible with many other common database languages like MySQL, PostgreSQL, Microsoft SQL Server, and Oracle. This will allow for smooth transitions if it is ever necessary to change the current database structure to a higher level software due sqlitebrowser’s export functionality. The database currently only has 4 tables: UserTable, ProjectTable, TaskTable, and ProgressTable. These 4 tables hold all the key data that the app will use on a daily basis. Due to the database’s mutable nature looking for ways to continually improve its speed and lessen its storage complexity will be an easy task.

**2.2 Type of Database**

The type of database used in the Project Management App is a relational database powered by SQLite. Currently, It has a total of 4 tables. Each table is essential for application use. The tables illustrated are as follows: UserTable, ProjectTable, TaskTable, ProgressTable. They will be in the next section.

**2.3 Tables and Attributes**

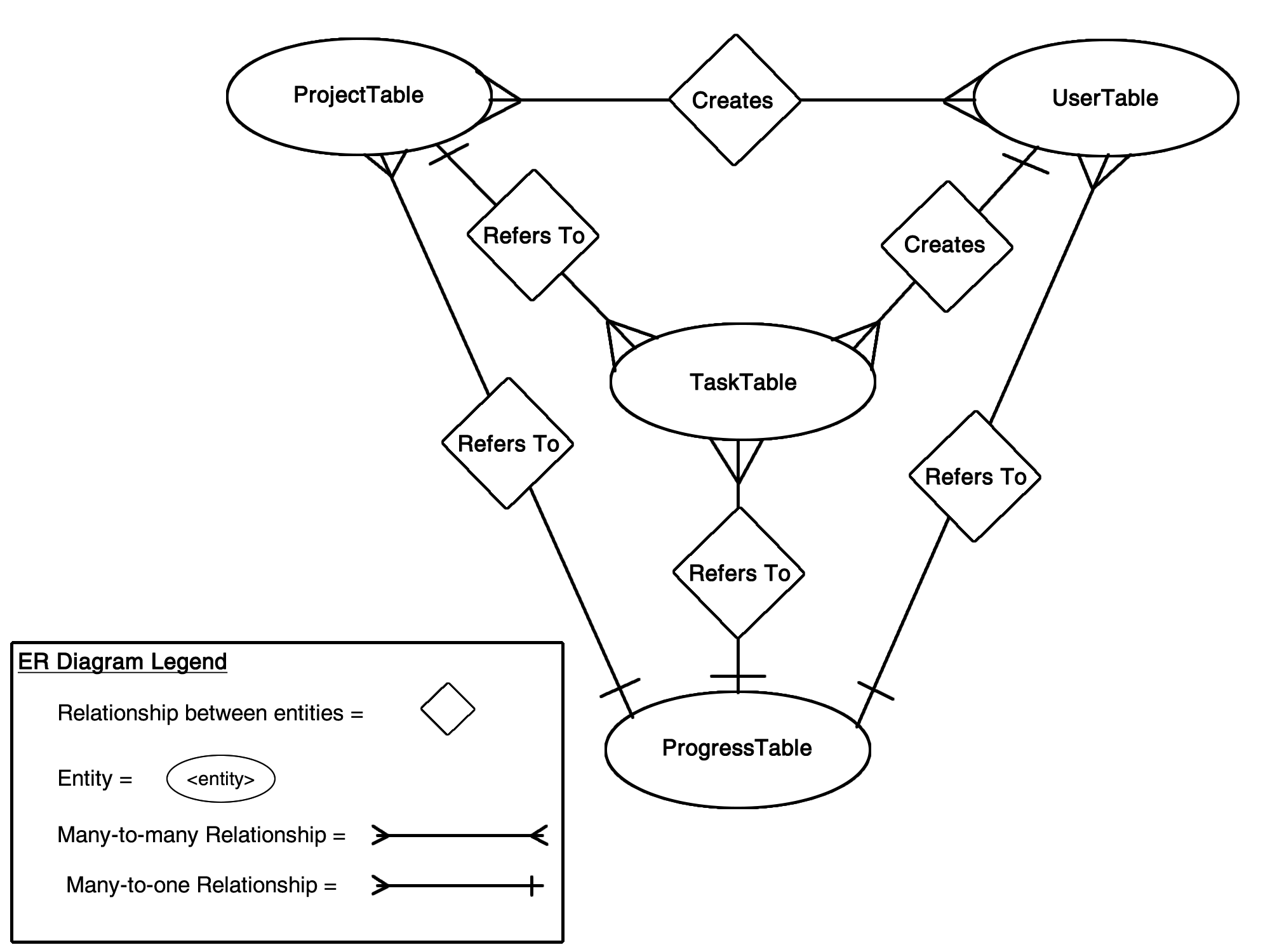
Table 0 shows all necessary database tables and their corresponding attributes.

**Table 0: Tables in the database**

|  |  |
| --- | --- |
| Table | Attributes |
| UserTable | UserID (primary key), FirstName, LastName, Email, Password, Bio, ProjectList, Picture |
| ProjectTable | ProjectID (primary key), Leader, MemberList, TaskList, ProjectName, ProjectDescription, ProjectProgress |
| TaskTable | TaskID (primary key), User, Project, TaskName, TaskDescription, TaskProgress, TaskStatus, TaskPriority, TaskDueDate, TaskDependency |
| ProgressTable | ProgressID (primary key), User, Project, Task, AllTaskProgress |

**2.4 Relationships**

In this section, an Entity-Relationship, or ER Diagram, is used to show each table and its relationship to other tables. Element from the UserTable can create projects in the ProjectTable and tasks in the TaskTable. The ProgressTable refers to the overall progress of a user for all the tasks they are assigned to for all their projects. (This diagram was made using Adobe Fireworks CS6)



**2.5 Table Details**

This section contains a brief description of each table and a chart, documenting its attributes, attribute types, and attribute descriptions. Please note that this relational database is using SQLite. Therefore, it is very light weight and does not come with all the usual bells and whistles of MySQL. For example, the data type VARCHAR(255) falls within SQLite’s TEXT affinity, this is why you won’t see the data type size in the tables to follow. For more information on this topic please consult <https://www.sqlite.org/datatype3.html>.

**2.5.1 UserTable**

The UserTable (See Table 1) is used to hold the basic information of a system user. The UserID acts as the primary key and is automatically generated and incremented each time a new user is created. The only required fields (designated by a \*) for creating an account are Email and Password to allow the user to quickly access the application. Users will be able to update the fields: FirstName, LastName, Bio, and Picture from the profile screen within the app. Lastly, the ProjectList column will hold a comma separated string of ProjectIDs. This section is multi-valued to allow for easy access to all projects a specific user is a part of.

**Table 1: User Table**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description & Example |
| UserID | INTEGER | This column is the primary key and is the unique auto incremented row identifier. Example: 1 |
| FirstName | TEXT | This column holds a string variable of the user’s first name. Example: Tyler |
| LastName | TEXT | This column holds a string variable of the user’s last name. Example: Mariano |
| \*Email | TEXT | This column is a required field that must also be unique so users cannot have the same email. Example: tmari795@live.kutztown.edu |
| \*Password | TEXT | This column is a required field to hold the password for which the storage specification are to be determined. Example: Mariano2365 |
| Bio | TEXT | This column holds a string which contain a user’s skills, title, experience, etc. Example: System Designer for 3 years |
| ProjectList | TEXT | This column holds a comma separated list of ProjectIDs from the ProjectTable. Example: 1,2,3,4,5,6… |
| Picture | BLOB | This column used to hold image URL. Example: http://104.238.131.94:5000/img.jpg |

**2.5.2 ProjectTable**

The ProjectTable (See Table 2) is used to hold all the information of a project. The ProjectID acts as the primary key and is automatically generated and incremented each time a new project is created. The required fields (designated by a \*) for creating a project are Leader, ProjectName and ProjectDescription which can all be found on the create account screen. The Leader field will automatically be filled in with the UserID of the project creator and acts as a foreign key. The leader will be able to update the fields: MemberList and TaskList from the leader view screen within the app. Lastly, the MemberList and TaskList columns will hold a comma separated string of UserIDs and TaskIDs. These sections are multi-valued to allow for easy access, for the leader, to all project members and all project tasks contained within a specific project.

**Table 2: Project Table**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description & Example |
| ProjectID | INTEGER | This column is the primary key and is the unique auto incremented row identifier. Example: 1 |
| \*Leader | TEXT | This column is required and is a foreign key relating to the project creator’s UserID from the UserTable. Example: 1 |
| MemberList | TEXT | This column holds a comma separated list of UserIDs from the UserTable. Example: 1,2,3,4,5,6… |
| TaskList | TEXT | This column holds a comma separated list of TaskIDs from the TaskTable. Example: 1,2,3,4,5,6… |
| \*ProjectName | TEXT | This column is a required field that holds a string variable of the project’s name. Example: Flappy Bird |
| \*ProjectDescription | TEXT | This column is a required field that holds a string variable of the project’s description. Example: Create a mobile game |
| ProjectProgress | REAL | This column is a field that holds a real value of the project’s progress based on positively reviewed task submissions of the overall project. Example: 33.3 |

**2.5.3 TaskTable**

The TaskTable (See Table 3) is used to hold all the information of a task. The TaskID acts as the primary key and is automatically generated and incremented each time a new task is created. The User and Project fields will be foreign keys to allow the task to relate to the ProjectTable and the UserTable so that the task knows which project the task belongs to and who is assigned to complete it. The required fields (designated by a \*) for creating a task are User, Project, TaskName, TaskDescription, TaskPriority, and TaskDueDate which can all be found on the create task screen via the leader screen. The leader will be able to update the fields: TaskStatus and TaskProgress from the leader view screen within the app after a task has been submitted.

**Table 3: Task Table**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description & Example |
| TaskID | INTEGER | This column is the primary key and is the unique auto incremented row identifier. Example: 0 |
| \*User | INTEGER | This column is required and is a foreign key relating to the UserID from the UserTable. Example: 1 |
| \*Project | INTEGER | This column is required and is a foreign key relating to the ProjectID from the ProjectTable. Example: 1 |
| \*TaskName | TEXT | This column is a required field that holds a string variable of the task’s name. Example: Design User Interface |
| \*TaskDescription | TEXT | This column is a required field that holds a string variable of the task’s description. Example: Design a visually appealing interface for the Flappy Bird project |
| TaskProgress | REAL | This column is a field that holds a real value of the task’s progress based on positively reviewed task submissions. Example: 33.3 |
| TaskStatus | TEXT | This column is a field that holds a string variable of the task’s status once it has been submitted. Example: Accepted, partially accepted, or failed |
| \*TaskPriority | INTEGER | This column is a required field that holds an integer value of the task’s priority. Example: Range from 1-5 |
| \*TaskDueDate | TEXT | This column is a required field that holds a string variable of the task’s due date in year-month-day format. Example: 2016-02-21 |
| TaskDependancy | TEXT | This column is a required field that holds a string variable of Yes or No. |

**2.5.3 ProgressTable**

The ProgressTable (See Table 4) is used to hold the task progress of all of a user’s tasks within all of their projects. For example, an individual task has it’s own progress. Whereas, a project has progress based on all of its assigned tasks. These tasks can be assigned to multiple project members. Therefore, the ProgressTable holds all the tasks that just one member is a assigned to within every project they are assigned. The ProgressID acts as the primary key and is automatically generated and incremented each time a new task is assigned to a new project member. The User, Project, and Task fields will be foreign keys to allow the progress to relate to the UserTable, ProjectTable, and TaskTable so the progress can differentiate between which tasks a user has for only a specific project. Lastly, the AllTaskProgress field holds the sum of all of a user’s task progress within all their project.

**Table 4: Progress Table**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description & Example |
| ProgressID | INTEGER | This column is the primary key and is the unique auto incremented row identifier. Example: 0 |
| User | INTEGER | This column is required and is a foreign key relating to the UserID from the UserTable. Example: 1 |
| Project | INTEGER | This column is required and is a foreign key relating to the ProjectID from the ProjectTable. Example: 1 |
| Task | INTEGER | This column is required and is a foreign key relating to the TaskID from the TaskTable. Example: 1 |
| AllTaskProgress | REAL | This column is a field that holds a real value of a user’s total task progress in a specific project. Example: 33.3 |