DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF NEBRASKA—LINCOLN

TODO List

CSCE 361 – Design Document

Group 4: Junzhe Cai, Zixuan Hao, Guodong Xue, Haoshu Zhang

2/23/2017

Contents

1. Introduction

The purpose of this design document is to demonstrate the high-level architecture, class diagrams, and database component for the TODO List web-based application. In the first section, this document will include architecture of the system, consisting of one architectural diagram, and text description of this diagram. In second section, it will include class diagrams for the system, which will display how the system is decomposed into classes, what methods in those classes, and how those classes relate to others. In third section, it will include database component, which will describe the structure and contents of data. The audience of this document is software engineers and system architects who will develop this system.

2. Architecture

Architecture establishes the context for further design and implementation, and it focuses on significant design decisions that have a lasting impact on the performance, reliability, cost, and resilience of the system.

2.1 Introduction

The high level architectural design of TODO List System will be layered systems pattern. The lowest level is the database layer, and the database is going to store the information of users and tasks. The data access layer can convert the data into a format that can be used in logic layer and GUI layer from database level through Redux. The logic layer, written in React, will provide functions on managing data and connecting with GUI level. The top level is the GUI level, users could access by browsers.

2.2 Modules

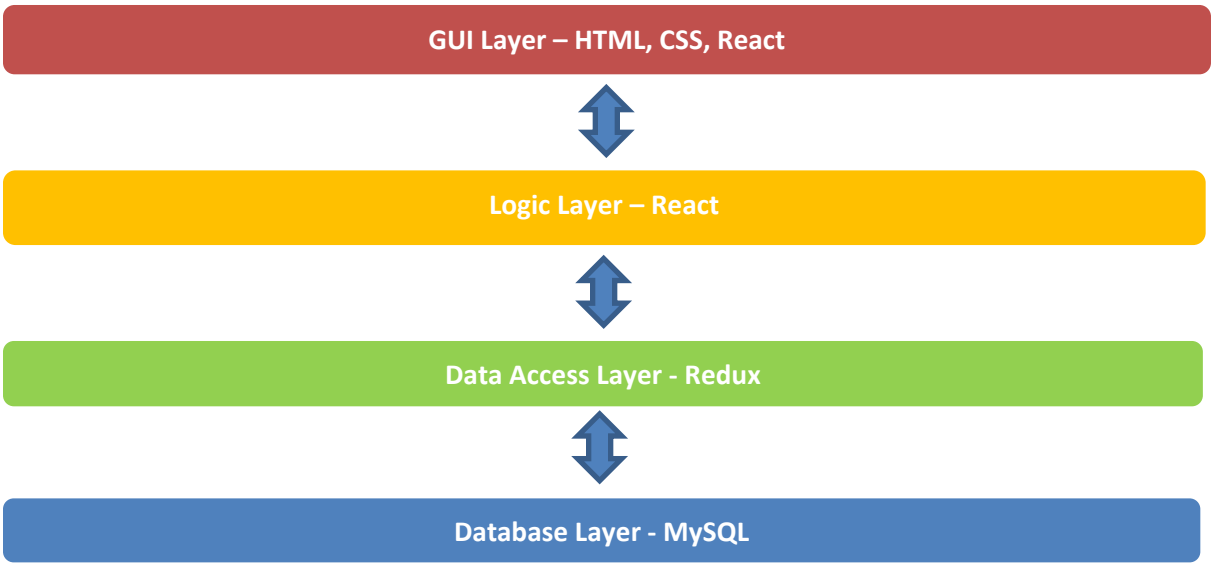


Figure 1: Layered Systems

2.2.1 Database Layer

The database layer is aim to store all persistent data in the system. For example, the database shall store the user's information, such as user name and email, from the registration function. The system will be using a MySQL database that can connect with data access layer easily by Redux. The data relationships are described more specific in section 4.

2.2.2 Data Access Layer

The data access layer is aim to convert the data into a format that can be used in logic layer and GUI layer from database level through Redux. The primary function of this layer is that accesses to the database and stores the data into the components of React in logical layer. It will also have a function that when user changes the data from GUI layer, it will store the changes after managed by logical layer.

2.2.3 Logical Layer

The logical layer is responsible for performing two main functions (induvial and group) to the React's components in the system. This layer will manage the data from data access layer and support GUI layer to display. Meanwhile, it will also be required to handle the changes from GUI layer made by users.

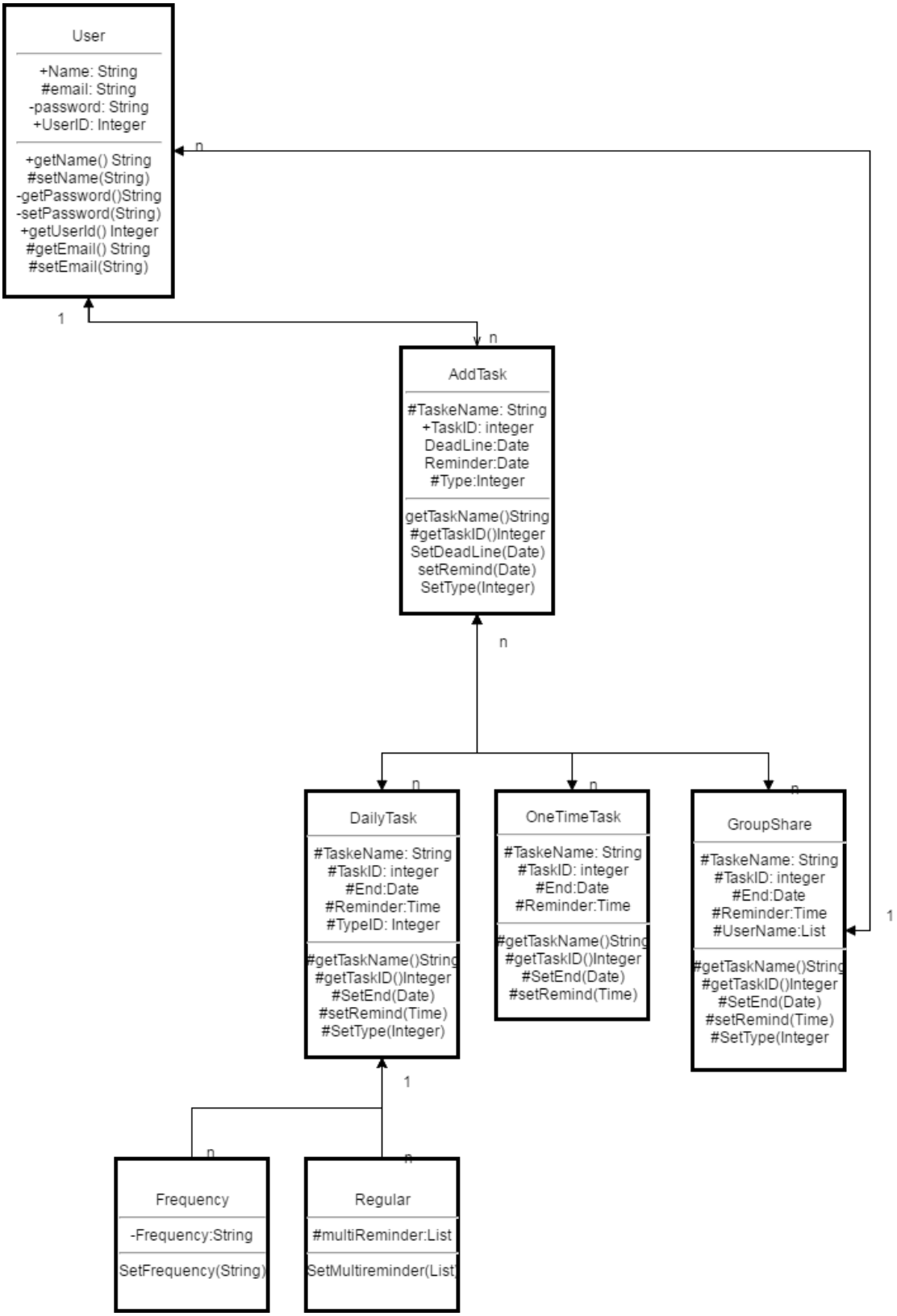
2.2.4 GUI Layer

GUI layer is the primary layer that users will complete their operations. In this layer, user could add, delete, and modify tasks. User could register for complete functions of system, the group function and email reminder.

3. Class Diagrams

In this section, it will show how the system is decomposed into classes, what methods in those classes, and how those classes relate to others.

3.1 Schema of data table class diagrams



3.1.1 Schema Information

User: Holds a data for a single user of the system including information, user name, user ID, email, and password.

AddTask: The table will represent the tasks edit by user, each task follows with the ID and name created by users. Each task follows with its deadline and remind time set by user. There are types can be chosen by user follows with integers.

DailyTask: The table will represent the tasks that is daily task type chosen by user. Each task created by user could set the two different types. User shall be able to set the name for the task and deadline and remind time.

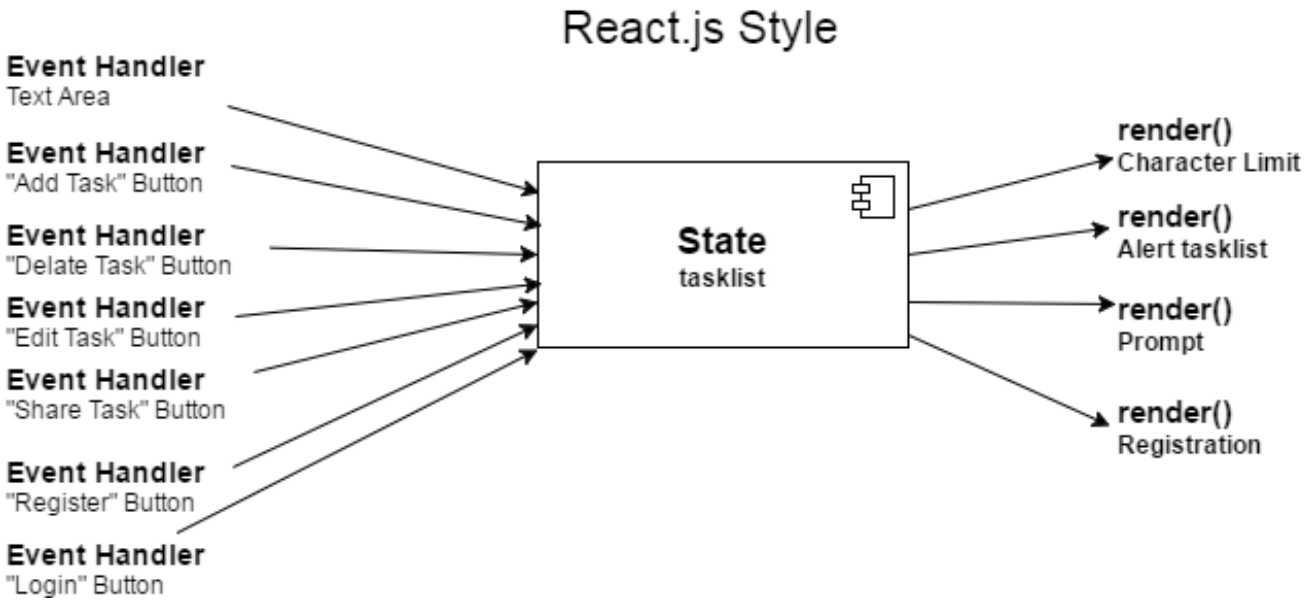
OneTimeTask: The table will represent the tasks that is one time task. It can set the deadline and remind time.

GroupShare: The table will represent the tasks that are tasks can be shared with other users. User could set who the users would like to share with.

Frequency: The table will represent the tasks that is several times of tasks. User could set how many times they would like to have the task on the schedule.

Regular: The table will represent the tasks that is regular tasks user is doing every day. For example, remind of car locking, taking the keys, etc.

3.2 GUI Layer Schema



3.2.1 Schema Information

The top layer of the system consists only one page displayed on browser. When a user click "Register" Button, the registration window is presented. When a user click "Login" Button, the login window is presented.

Event Hander: The GUI layer handles user's operations, such as pressing button and type words.

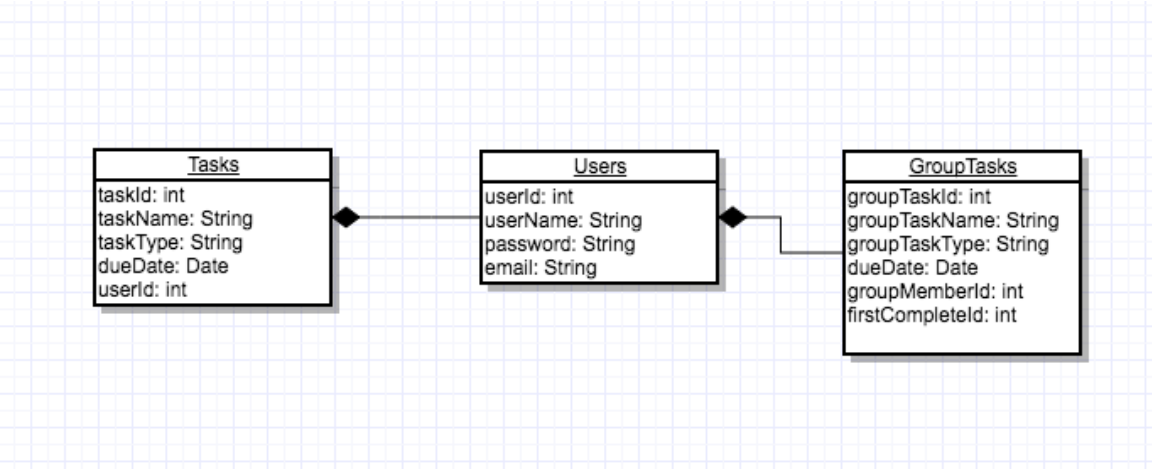
Render(): Render a React element into the DOM in the supplied container and return a reference to the component

4. Database component

4.1 Introduction

The database can be divided into three tables: task table, user table and group. Each table will handle different situation of the application.

4.2 Diagram of database



4.3 Database tables

Task	
Element	Type
Task ID	int
Task Name	String
Task Type	String
Due Date	Date
User ID	int

This table is used to store the data of the users create for their personal tasks. It is used to handle the basic function of this application.

Users	
Element	Type
User Name	String
Password	String
Email	String

This table is used to store the data of the users' registration information such as user id, password and their email address. It is used for the users to log in to the TODO list.

Group Tasks	
Element	Type
Group Task ID	int
Group Task Name	String
Group Task Type	String
Due Date	Date
Group User ID	int
First Complete Id	int

This table is used to store the data of the group tasks such as how many people are involved in the group project and what is the due date of the task. It is used for several users who have the same tasks to challenge each other.

4.4 Relationships

These three tables are related to each other. If TODO list wants to send users notification of their personal tasks; it can go to the "task" table to see which task is near the due date. Then it can get the user name from the table. Use the user name, it can found the user's email in the "users" table to get their email address to send the notification.

If the application wants to tell other user who first completes the group tasks; it can find data inside the "group tasks" table using the group task ID and first complete user id. Then it can get the user name of other users. After that, it can get the email address of these users to tell them who complete the group task first.