

Aaron Barbary, Alex Luongo, Daniel Wallace, Jordan Strong, Joshua Golding, Matthew Czech
Dr. Baliga
Senior Project
October 15, 2017
Task Manager Software Project
Design Document

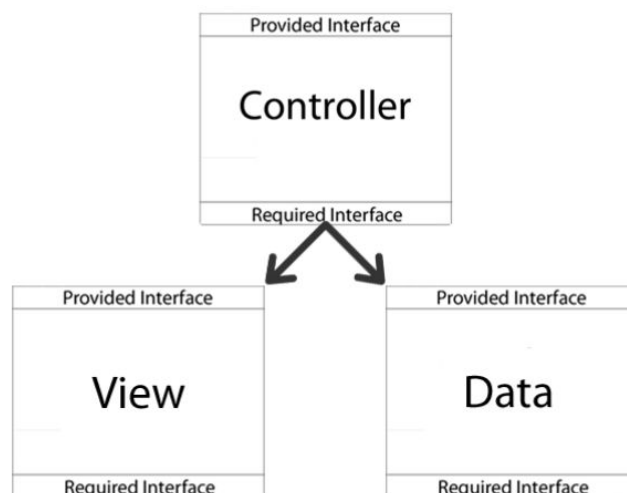
1. Introduction

This is the design document for the Task Management application, per the description in the Project Specification. This program will include the following capabilities:

- User accounts (admin or regular) and authentication
- A database that stores the users, as well as tasks and subtasks
- The creation of tasks by both admin and regular users
- Delegation of tasks and subtasks by users to other users
- Automatically determine the best user to assign a task based on number of tasks users have in their queue
- Assigning tasks to multiple users

2. Architecture

This high-level architecture shows the hierarchy the program will use. It will have a controller that is split in a client-server model. The server will be responsible for handling data to and from the database and to and from the client, and the client will update and receive input from the GUI, as well as send and receive data requests to the server. The GUI will be used to display data from the database, as well gather data from the user to store in the database.



3. Modules

3.1 Controller

3.1.1 UI Controller

The UI Controller module will be responsible for taking the actions from the GUI and sending them to either the Load Database or Task Object Builder modules. It will also be responsible for UI functions that do not require the database.

ArrayList<Task> loadUserTasks(int userID)

Definition:

Retrieves all of the tasks in the database assigned to the logged in user

Parameters:

int userID: the userID assigned to the logged in user

Returns:

An ArrayList of Task objects containing the tasks assigned to the user

Task addTask(String projectNum, String name, String dateDue, String assignedUser, String description, String notes)

Definition:

Add a task to the database with the given parameters

Parameters:

String projectNum: number of the project the task was created for

String name: The name of the task

String dateDue: The date that the task is due by

String assignedUser: The user that the task is assigned to

String description: A detailed description of the task

String notes: Miscellaneous details about the task that don't fit in the description

Returns:

The created Task object

3.1.2 Task Object Builder

Accepts as input the information needed to build a task object, from either the UI or the database. When from the UI, it takes the information entered by the user then passed through the UI controller, and uses it to output a Task object. When from the database, a request has been made from the UI for information from the database. Task Object Builder will take the results of the query and use it to build a Task object, then pass it back towards the UI.

Task createTaskObject(String projectNum, String name, String dateDue, String assignedUser, String description, String notes)

Definition:

Creates a Task object with the given parameters

Parameters:

String projectNum: number of the project the task was created for

String name: The name of the task

String dateDue: The date that the task is due by

String assignedUser: The user that the task is assigned to

String description: A detailed description of the task

String notes: Miscellaneous details about the task that don't fit in the description

Returns:

The created Task object

3.1.3 Load Database

Load Database is the module that is used to call all of the info relevant to the user (e.g. all tasks assigned to the user). It is used when the user first logs in, and at regular intervals to update the client.

ArrayList getTasks(int userID)

Definition:

Retrieves all of the tasks in the database assigned to the logged in user

Parameters:

int userID: the userID assigned to the logged in user

Returns:

An ArrayList of Task objects containing the tasks assigned to the user

3.1.4 SQL Query Builder

SQL Query Builder is responsible for translating requests for information from the database into SQL queries. It will be capable of translating requests for both adding to the database and pulling data from the database.

Task getTask(int taskID)

Definition:

Retrieves a specific task by the ID assigned to the task

Parameters:

int taskID: the ID of the task to be retrieved

Returns:

A Task object containing the requested task

ArrayList getAllTasksForUser(int userID)

Definition:

Creates a SQL query to retrieve all of the tasks in the database assigned to the logged in user

Parameters:

int userID: the userID assigned to the logged in user

Returns:

An ArrayList of Task objects containing the tasks assigned to the user

int addTask(Task task)

Definition:

Creates a SQL query to add the Task object to the database

Parameters:

Task task: The Task object to be added to the database

Returns:

Whether or not task was added successfully

3.1.5 Task

The Task class is used to create Task objects, which will be used to hold all of the

Information about a single task. Tasks will

Task(String projectNum, String name, String dateDue, String assignedUser, String description, String notes)

Definition:

Constructor for Task class

Parameters:

String projectNum: number of the project the task was created for

String name: The name of the task

String dateDue: The date that the task is due by

String assignedUser: The user that the task is assigned to

String description: A detailed description of the task

String notes: Miscellaneous details about the task that don't fit in the description

Returns:

A Task object containing all of the information that was input

String getprojectNum()

Definition:

Returns the task's projectNum field

Parameters:

None

Returns:

The task's ID number

String getName()

Definition:

Returns the task's name field

Parameters:

None

Returns:

The task's name

String getDateDue()

Definition:

Returns the task's dateDue field

Parameters:

None

Returns:

The task's due date

String getAssignedUser()

Definition:

Returns the task's assignedUserfield

Parameters:

None

Returns:

The task's assigned user

String getDescription()

Definition:

Returns the task's description field

Parameters:

None

Returns:

The task's description

String getNotes()

Definition:

Returns the task's notes field

Parameters:

None

Returns:

Any notes attached to the task

void setprojectNum(String projectNum)

Definition:

Sets the projectNum field

Parameters:

String projectNum: The task's number

Returns:

None

void setName(String name)

Definition:

Sets the name field

Parameters:

String name: The task's name

Returns:

None

void setDateDue(String dateDue)

Definition:

Sets the dateDue field

Parameters:

String dateDue: The task's due date

Returns:

None

void setAssignedUser(String assignedUser)

Definition:

Sets the assignedUserfield

Parameters:

String assignedUser: The task's assigned user

Returns:

None

void setDescription(String description)

Definition:

Sets the description field

Parameters:

String description: The task's description

Returns:

None

void setNotes(String notes)

Definition:

Sets the notes field

Parameters:

String notes: Any notes that the task has

Returns:

None

3.2 View

3.2.1 User Interface

The User Interface will contain a column on the left which displays the user's user name and profile picture at the top of the column and tabs that correspond to the various functions of the application. The tabs included in the left column are labeled tasks, Create New, Inbox(), Archive, Trash, Request Task, and Logout. The main UI takes input from the user to select one of the previously mentioned tabs. Once a tab is selected, the user can perform functions that correspond to the selected tab.

void MainWindow(int userId)

Definition:

Displays the program's main window

Parameters:

int userID: the logged in user's ID

Returns:

None

3.2.2 User Login

The User Login is a window that accepts input of a username and password. The password will be hashed and then compared to the password stored in the database for this user. If the password hashes being compared are the same, the user will have successfully been authenticated. After successful authentication, the main UI will be displayed.

3.3 Data

3.3.1 Database

The application will utilize a mySQL database for storing all information regarding the task managing process. This database will contain three tables to store information about tasks, sub-tasks, and the users for the application. Each task and sub-task entry will store its own unique identification value, a description of what the task requires, the time in which it was created, the user who created the task or sub-task, and the user who was assigned the task or sub-task. An entry in the sub-task table will store the unique identifying value of its parent task. An entry in the user table will store a unique identifying value, the username of the individual, the password of the individual, the user's first name, the user's last name, the number of tasks the user is currently assigned to, and the number of tasks the user has finished.

4. Integration Test Plan

4.1 Task Object Builder

Description

Check that the Task object is being correctly built

Input

Data for each individual field that will be stored in a Task.

Expected Output

A Task object that contains the parameters and data that the user input

4.2 Load Database

Description

This will test that Load Database pulls all of the tasks assigned to a given user, while not pulling any not assigned to that user

Input

User ID assigned to the user that is logged in

Expected Output

A request to be sent to SQL Query Builder to lookup all of the tasks in the database that are assigned to that user

4.3 SQL Query Builder

Description

This will check that the SQL builder is producing the correct SQL query to pull the necessary information from the database

Expected Output

From the Task Object Builder, a Task object that is to be added to the database

Output

A SQL query that will add the information in the Task object to the database

Input

A user ID from Load Database

Expected Output

A SQL Query that will return every task assigned to the user that corresponds to the user ID that was input

4.4 User Login

Description

This will test that the User Login outputs correct authentication when a user tries to login

Input

A username and password combination that exists in the database

Expected Output

Successful login

Input

A username and password combination that does not exist in the database

Expected Output

Authentication error

5. Prototype Description

The early prototype for this application consists of 2 main windows. When the application is started, a login window appears. This login window contains 2 text fields that allow the user to

enter their username and their password. The password text field hides the text being entered by the user. The login screen contains a login button that currently opens the main window without any authentication. When the main window is opened, multiple tabs are located on the left side. When selecting the “Tasks” tab, a pane becomes visible in the main window with 2 tabs. The first tab is labeled “My Tasks”, which will show a list of the user’s tasks. The second tab is labeled “All User Tasks”, which will show a list of all users’ tasks. Each tab will display a table of the respective tasks, which contains each individual field of that task. The fields of that task will be displayed in separate columns.

When selecting the “Create New” tab, a pane with multiple text fields and 2 buttons becomes visible in the main window. Each text field has a label that specifies what information needs to be entered into each field. When clicking the “Create” button, a check will be performed to see if the user entered a task name. If the user did not enter a task name, a JOptionPane appears in the center of the screen stating that a task name must be entered before a task can be created. The user can click the “OK” button to return to the “Create Task” pane. If a User enters a task name and/or any other information for a task and clicks the “Create” button, the Main Window will transition its display to the “My Tasks” tab within the “Tasks” pane and display the updated table showing the newly created task and all of its fields. If the user clicks the “Cancel” button at any time, all of the text fields will be cleared and the main window will transition its display to the “My Tasks” tab within the “Tasks” pane.

When selecting the “Inbox()”, “Archive”, or “Trash” tabs, a pane displaying the table header for task fields will appear. When selecting the “Request Task” tab a pane will appear containing a “Request Task” button and a drop down field. The drop down field does not currently display any options. However, during the implementation phase, a list of all users will be added to the drop down field and the user will be able to select any individual user to request a task from when clicking the “Request Task” button.

When selecting the “Logout” tab, a pane will appear containing a “Logout” button, which currently has no functionality when clicked. The goal during the implementation phase is to have the logout button on the left side of the screen. Upon clicking the logout button, it is anticipated to return to the login screen.

6. Project Plan

The Project Plan lays out goals for the first 3 weeks of the implementation phase, as well as the tasks assigned to each member of the group. Within the next 3 weeks, we plan to accomplish the following:

1. Add user authentication functionality by storing user data in the database (username, password hash). Hash the password entered when the user clicks the “Login” button on

the login page and compared it to the hash of the password stored for that user. If they match, the main window will appear.

2. Finalize the fields that need to be entered when creating a task. It is possible that new task fields will be necessary. It is also possible that removing/re-identifying certain task fields will be necessary. All task fields will still be able to be displayed within the tables in the "Tasks" pane.
3. Add inbox functionality for a user. When a user is sent a task or a task request, it will be displayed in a table in the "Inbox()" pane. It is possible that 2 tabs labeled "Task Requests" and "Tasks Pending/Received" will be needed within this pane to display a table of the respective task objects. Functionality will also be added to accept/deny a pending/received task.
4. Add functionality to the "Archive" tab to display a table of task objects that have been fully completed by a user. No in progress tasks will be visible in this table.
5. Add functionality to double click on tasks that are displayed in tabled in order to edit their fields.
6. Add functionality for the user to be able to edit a task's completion status/progress. When a task is fully complete, it should be automatically sent to the "Archive" table and should not be visible within the "Tasks" pane.
7. Determine whether the "Trash" tab is needed or not. It may prove useful to keep deleted tasks temporarily, but it also may be unnecessary as a task object does not contain much data and can easily be recreated.
8. Add functionality to the "Logout" button so that a user is logged out and the login window reappears.
9. Add functionality to sort tasks within any task object tables by any task field accordingly.
10. Add functionality to search tasks for information entered by the user.
11. Clean up the layout of the GUI for this application so that everything is displayed in a more organized fashion.
12. Modernize the GUI so that the graphics look professional and so that we look like professional developers.

Below are the responsibilities for each group member developing this task management application.

- Aaron Barbary - The manipulation of data extracted from the database via queries to produce specific lists of data from the database.
- Alex Luongo - Overseeing and contributing heavily to the construction/modernization of the layout of the GUI and contribute to the functionality of the GUI.
- Daniel Wallace - Modernizing the appearance of the GUI for a more professional look. Contributing to the functionality of the GUI. Working on developing searching algorithms/sorting algorithms so that users can search tasks for specific information and sort tasks by task fields.

- Jordan Strong - Assisting in construction of the layout of the GUI, adding functionality to the GUI, and working on the communication between a user's machine running an instance of the application and the server containing the database information.
- Joshua Golding - Contributing to the design and management of mySQL database. Contributing to modernizing the layout of the GUI.
- Matt Czech - Responsible for the design and management of mySQL database including all tables and columns.

Each member of the team developing this application will be working to accomplish the 12 objectives of the project plan detailed above. It is anticipated that more functionality/layout design will be required as new ideas for this application are presented.