Track List

Complete Documentation

CSCI 201 – Principles of Software Development

Team: Prateek Bhatia, Shawn Fahimi, Mi Hyun An, Jiaxin Cheng, Tianmu Lei, Jeffrey Nagel

Class: CSCI 201L

Table of Contents

Overall Concept

This desktop application provides a platform for users to coordinate productivity in a group project among the group members by creating a shared To-Do List. Any group member can create a list, shared it with other users, and add tasks to the shared list. The application allows users to specify a deadline for each task and displays the amount of time remaining to deadline in the Timer Mode feature. Tasks are also personalized for users in the MyWorklist feature of the application, which displays all the tasks created by a user across all of the user's To-Do Lists. Users receive notifications when a list is shared with them, and/or when tasks are added to a project they are a part of. Users can mark tasks as completed, which sends a notification to other group members informing them of the task completion.

Overall, this application helps groups better manage time and productivity in a project by organizing their tasks into mutually-shared lists, which helps them track group as well as individual progress on different components of the project.

High Level Requirements Document

Our team's project will consist of a desktop application that serves to help any group of users better organize the tasks they may need to complete between themselves in a given timeframe, while also coordinating specific task management for individual users (similar to the to-do application Wunderlist).

Login Screen: Upon startup, the user must be greeted by a small login screen that will prompt the user to enter a username and password. The user can either choose to login with their unique username and chosen password, if they already have an account on the application, or use the information to sign up and create a new account if they don't already have one. There is also the option to sign in as a guest to the application. The only features guest accounts may use/manipulate are the To-Do List and Notifications (personal task reminders). These features are specified later in the document.

Main Menu: Once logged in, the user will be taken to the main menu of the application, which will provide the user with the option to access any of the following application components: To-Do List, My Worklist, Notifications, and Timer Mode.

The To-Do List: The To-Do List is the heart of the application, where all of the group's tasks are stored. At the top of the To-Do List is the option to add new tasks, which will be denoted by a "+" sign in the upper right corner of the list. When adding a task, a specified date and time of task deadline must be added along with a very brief task description ("Task ABC, due 11/19 at 2:45" would be considered a complete task entry). All tasks in the To-Do List are sorted in the order in which they need to be completed (task priority of Earliest Deadline First). Users can share their To-Do Lists with other users, who can now receive notifications and see the tasks in that To-Do List. The To-Do List is one of the core components of the application's networking abilities, as it allows any group of users to coordinate task responsibility between its members. As stated earlier, this is the only feature Guest accounts have access to.

My Worklist: My Worklist displays a list of the logged in user's personalized task load (which includes only tasks created by the user in question), sorted by Earliest Deadline First.

Notifications: The Notifications tab contains personal task reminders, notifications on joining a group's To-Do List, and notifications of newly added tasks. The Notifications tab is one of the facets of the applications that allows networking with other users.

Timer Mode: Timer Mode is a button in the My Worklist tab. When clicked, the timer mode show the list of the tasks sorted by Earliest Deadline First, however with a timer next to each task that counts down to the task's due date. Additionally, there is an alarm feature, which notifies the user with a beep after a certain time period, which is specified by the user through a JTimeChooser adjacent to the timer mode button.

Technical Specifications

Database

- Creating Tables (3 hours)

  PrototypeTableName - PrototypeColumn1, PrototypeColumn2, …

1. *UserAccount* - Username (Primary Key), Password, First Name, Last Name
2. *Notification* - NotificationID, Message, Owner, MarkRead
3. *ToDoList* - ListID, Name, Owner
4. *ToDoListMember* – MemberID, MemberName, ListID
5. *Task* – TaskID, TaskInfo, CreatedBy, ListID, TaskDeadlineYear, TaskDeadlineMonth, TaskDeadlineDay, TaskDeadlineHour, TaskDeadlineMinute, Completed

- Login (3 hours)
  - On receiving a request from the server with a username and password, check in the SQL Database if the username and password combination exists, and send the response (true/false) back to the server
  - After verification, send all of the user's data from SQL Database to the server to populate the To-Do List tab, My Worklist tab and Notifications tab for the user
- Sign Up (3 hours)
  - On receiving a request from the server with a username and password, check in the SQL Database if the username exists. If it does, inform the user and ask him to choose another username. Otherwise, add the username and password to the SQL Database.
- Guest Login (2 hours)
  - This does not make any changes to the Database. Any actions performed in the Application by a guest would be saved locally, but not to the server database.
  - When a guest user logs out, all data would be lost. This gives the guest a look at the working of the application in a demo mode but limits the functionality.
- To-Do List (5 hours)
  - Whenever a new To-Do List is created
    - check if To-Do List name is available
    - if available, the To-Do List Name and Creator Username are added to the SQL Database under the ToDoListMember table associated
  - Whenever a new Task is added to an existing To-Do List
    - The task is added to the ToDoList table (with details like Task Deadline, Assigned To, Assigned By), and Notifications table (with details like Task Deadline, Assigned Time

- My Worklist (3 hours)
    - Whenever a new Task is created, the Task and related details along with the Username of the user who created the task are added to the table
- Notification (3 hours)
    - Whenever a new Task is created, the corresponding Notification, Username of the user who created the task, and the date and time of assignment are added to the table, and the user is notified of the assignment through the Notification tab

Client

- Login (3 hours)
  - If the GUI calls the method receiveUserData to send a combination of username and password to the client when someone is trying to login, the client will serialize the account class and send the serialized object to the server.
  - After receiving String indicating the state of the account from the server, the client will call another method for the GUI to receive the information about the combination of the username and password.
- My Work List (4 hours)
  - When the user wants to see the content of my work list, the client will send an Object to the server to request the work list for this user.
  - Once the client receives the content from the server, the client will call a method in GUI so that GUI can get the information needed.
  - If the user marks a task as ‘completed’, a message is sent to the server so that the server can mark the task as ‘completed’ in the database, and the client will call a method in GUI so that GUI can update the content of the user’s work list.
- To Do List (3 hours)
  - If the GUI request for a “To Do List” of a user or the content of a specific list, the client will sent the request to the sever. After the client receives the information, from the server, the client will call a method in GUI so that GUI can get the information needed.
  - If the GUI tells the client that a user creates a task The client is going to send the information to the server. After this process is done, we request to server for the new content for this specific work list so that the GUI can be updated.
- Timer (3 hours)
  - If the GUI calls the method startTimeMode in the client, the client will calculate the remaining time to the deadline and start countdown and update with the time with the GUI every minute to show the change.
  - When the countdown goes to zero, client will call a method in GUI to alert the user.
- Notification (5 hours)
  - When the client calls the method openNotifications when the user clicks on the Notifications tab, we request server the content of Notifications for the user and then will send the content to the GUI

- When the GUI call a method if the user clicks the accept button for any invitation they received, the client will send to the server that adding the user to the group and add tasks to their work list.

Server

- Server Infrastructure (5 hours)
  - Will create a base server using a ServerSocket
  - Will hold a vector of threads (ServerThread) which will be created with each new client connection
  - If a client attempts to establish a connection, the server will temporarily accept and wait for an incoming account object.
    - If the object is a login request, the server will query DB to check that the user exists, and then send the client his personal items and lists.
    - If a create account request, the server will make sure that the account does not already exist in the DB, then create the account in the DB and authenticate the user
  - The server will share the same backend objects as client and will handle deserialization and casting before obtaining data from them
- ServerThreads (5 hours)
  - Each ServerThread will have an ObjectInputStream and ObjectOutputStream to send and receive serialized objects from the client. A special case of sending and receiving objects is if two threads correspond to the same account open on different devices. This will be handled appropriately and everything will be synchronized based on account
  - Based on what the content of the received object is, the thread may modify entries in the DB, query the DB for data, and send information to another thread if data is being shared among users
  - If a user tags another user, the server will query the DB to check if that user exists, and if so, will send the according list and item to the user
- Server/Database Interface (5 hours)
  - The server and its respective classes will interact heavily with the DB. There must be a class which forms a connection to the SQL server at startup
  - This class will have multiple methods catered to the different get/post requests that the clients will make
  - Due to the small-scale nature of the project, only one server-database interface object thread will be necessary

User Interface

- Login (2 hours)
  - Login window has username and password text fields along with button to login, button to create account, and a button to login as a guest user
  - After authenticating an existing user or if the user chooses to create an account, the user will be directed to the main to do list application window and user information will be sent to the client
  - If the user does not exist, the login window will prompt the user to either login as a guest or to create an account
  - If the user chooses to login as a guest, then the user will be directed to the main to do list application window
- Tasks GUI (2 hours)
  - Each user input task will be shown with task information, tagged users, and deadline.
  - Whenever a new task is added, the information for that task will be sent to the client
- To-Do List (4 hours)
  - "To-Do List" feature is shown on the menu bar on the side of the main application window. In the menu bar, various lists created by the user will be shown under the To-Do List tab, which will be implemented as a tree. When user clicks on a particular list, the tasks for that particular list will be shown.
  - When the user clicks on the "+" button on the To-Do List tab in the menu bar, the user will be directed to an empty to do list panel, in which the user can add new tasks
  - If no To-Do List is shared with another user, then the to do list will remain as a personal list
- My Worklist (2 hours)
  - My Worklist is a feature shown on the menu bar on the side of the main application window
  - If the user clicks on "My Worklist", then the user will be directed to My Worklist panel which is filled with tasks that pertains to the current user

- Timer Mode (2 hours)
    - The timer mode is a JButton on top of the tasks in My Worklist. All the user's personal tasks from all of his groups are sorted by Earliest Deadline First. The user sets a time in a box right next to the Timer Mode button, and clicks on the button to start the timer mode. Then, all deadlines turn into 'time remaining until the deadline'. Also, when the set time goes to 0, an alarm is beeped.
- Notifications (2 hours)
    - The client will be provided a method that updates the Notifications panel
    - The notifications panel will contain invitations to join a shared to do list, notifications that a task has been assigned to the user, and notifications about the upcoming due date
    - The user will be able to go to the notifications panel by clicking on "Notifications" on the menu bar
    - If the notification is an invite, the user will be able to accept or reject by clicking an accept or a reject button shown on the invite
    - If the notification is a task has been assigned to the user or a message about the upcoming due date, the user will be able to clear those notifications by clicking on the clear button next to each notification

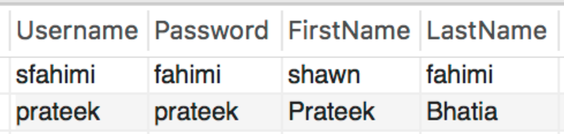
Detail Design Document

Database

<u>MYSQL TABLES</u>

UserAccounts

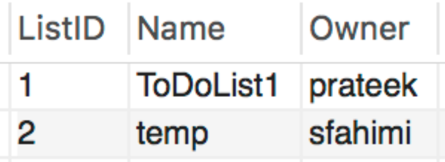
- Authentication - On a request from server, check in the table if username and password sent by the server exists in the table, and return true/false to the server
- Load all user data after authentication, and send the object back to the server
- Sign Up - On a request from server, check in the table if chosen username has not been taken, and return true/false to the server
- Add the user account to the table after unique username is chosen



Username	Password	FirstName	LastName
sfahimi	fahimi	shawn	fahimi
prateek	prateek	Prateek	Bhatia

ToDoList

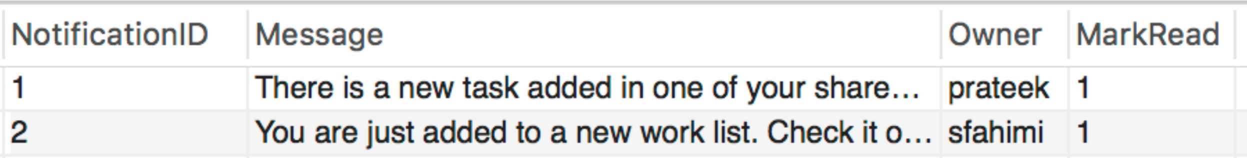
- On a request from server to add a new To-Do List, check if the To-Do List name does not already exist. If available, update ToDoListMember table
- On a request from server to add a new Task, if the To-Do List exists, add the task and related details to the table. Otherwise, further add the ToDoList to ToDoList table and ToDoListMember table
- Update the Completed column whenever a Task is marked completed
- UPDATE: There is no longer an AssignedTo column for Tasks. Tasks now contain additional data members that specify the deadline of a Task up until the minute (deadlineMonth, deadlineDay, deadlineHour, etc.).



ListID	Name	Owner
1	ToDoList1	prateek
2	temp	sfahimi

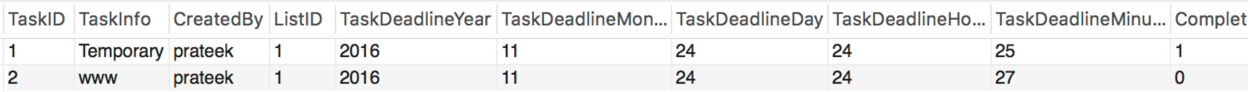
Notification

- Whenever a new Task is added, add the corresponding Notification and related details to the table
- Update the Read column whenever a notification is read by the assigned user
- UPDATE: There are no longer AssignedBy/AssignedOn* columns for Notifications.



NotificationID	Message	Owner	MarkRead
1	There is a new task added in one of your share…	prateek	1
2	You are just added to a new work list. Check it o…	sfahimi	1

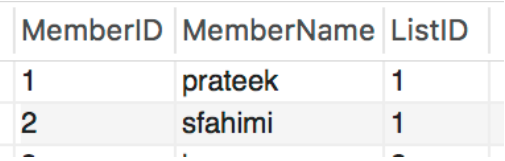
Task



TaskID	TaskInfo	CreatedBy	ListID	TaskDeadlineYear	TaskDeadlineMon...	TaskDeadlineDay	TaskDeadlineHo...	TaskDeadlineMinu...	Complet
1	Temporary	prateek	1	2016	11	24	24	25	1
2	www	prateek	1	2016	11	24	24	27	0

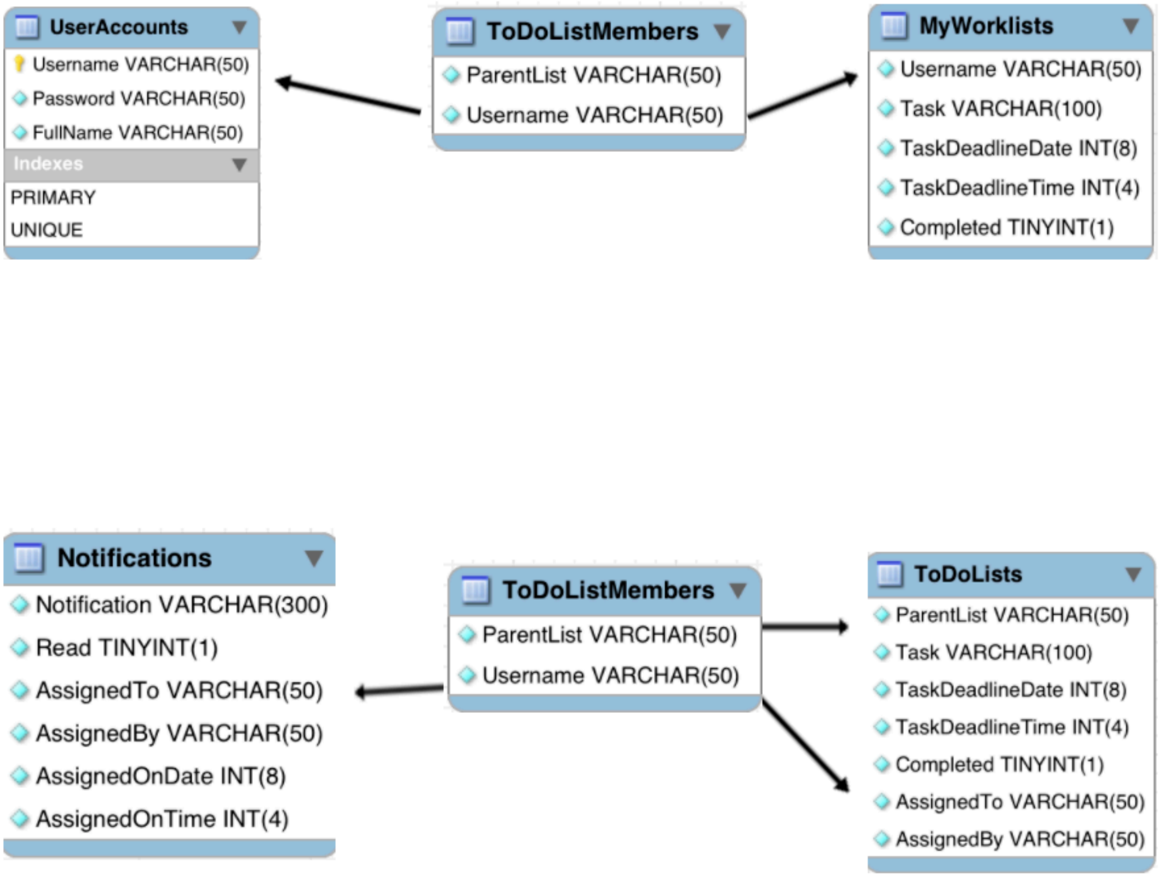
ToDoListMember

- Whenever a new user is joined to an existing To-Do List through a Task Assignment, add the Username and the corresponding To-Do List to the table
- Whenever a new To-Do List is created, add the Creator Username and the corresponding To-Do List Name to the table

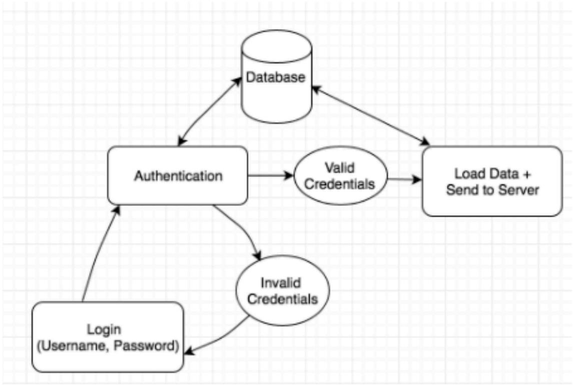


MemberID	MemberName	ListID
1	prateek	1
2	sfahimi	1

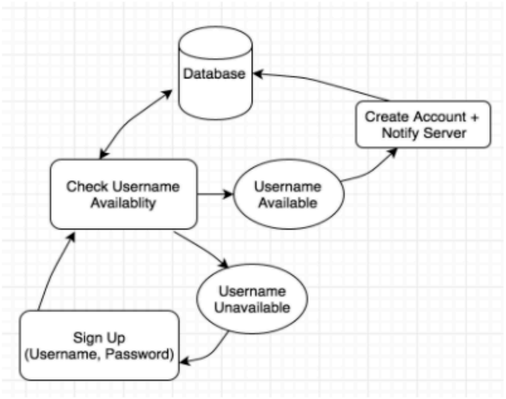
ER DIAGRAM (DATABASE)



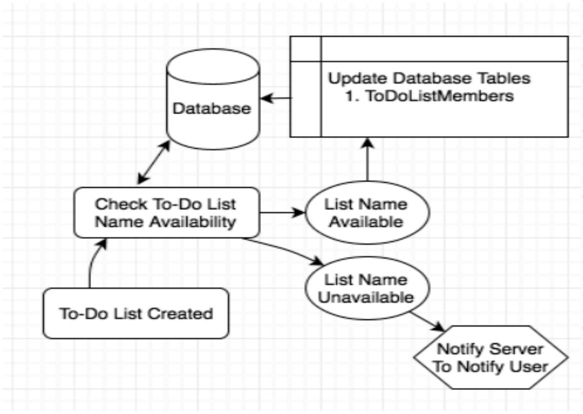
Login



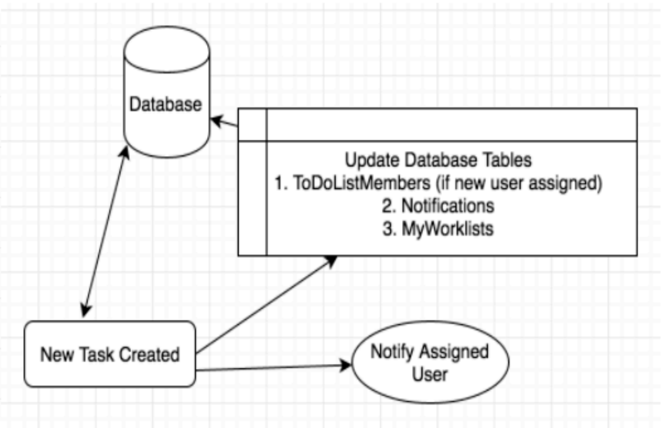
Sign Up


Database
Create Account + Notify Server
Check Username Availablity
Username Available
Username Unavailable
Sign Up (Username, Password)

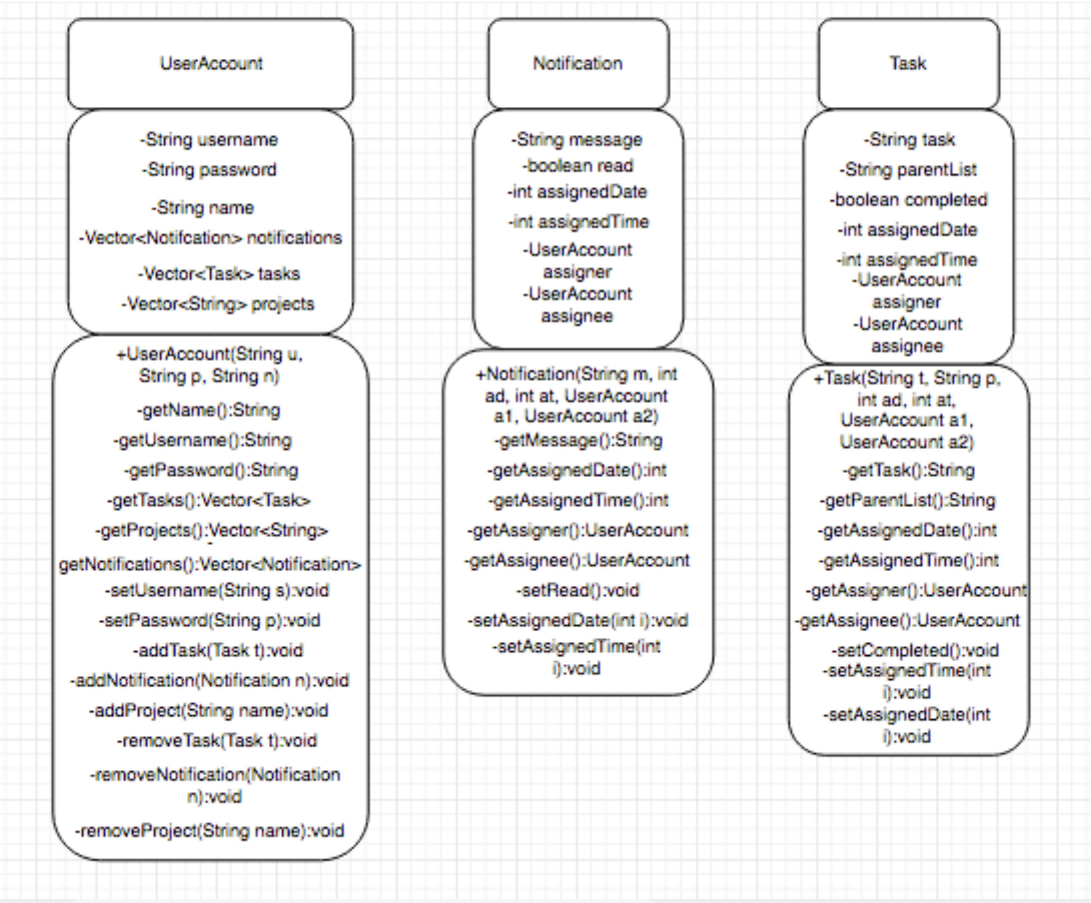
New To-Do List

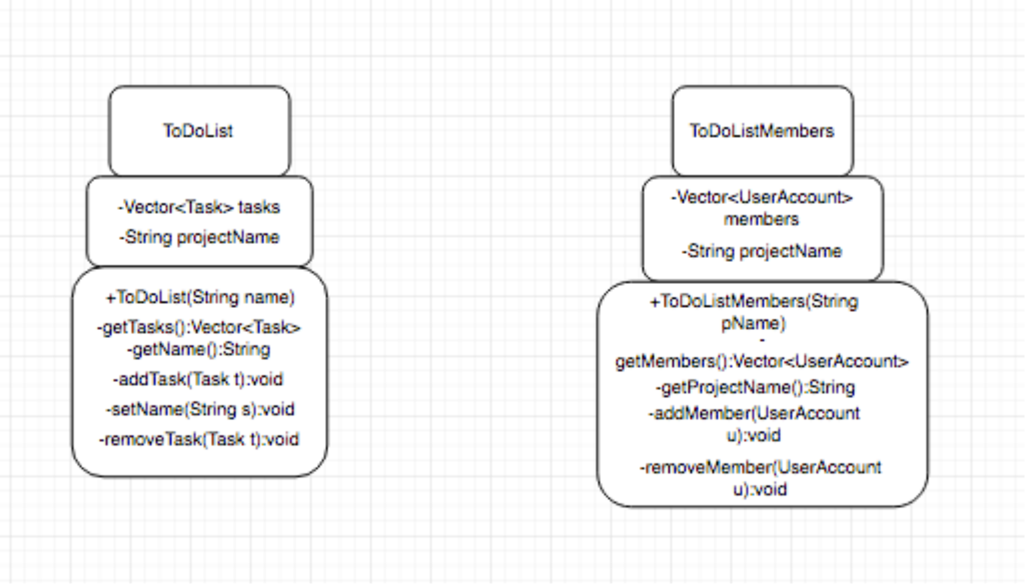


New Task/Notification



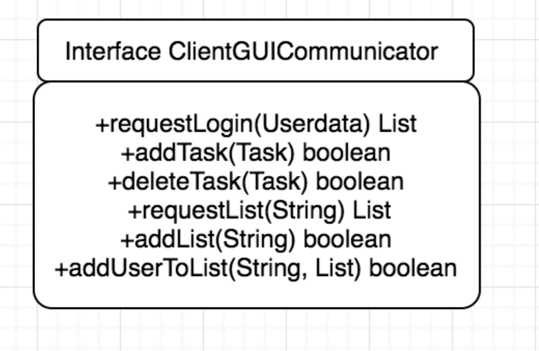
Class Implementations for the Application Database


UserAccount
-String username
-String password
-String name
-Vector<Notifcation> notifications
-Vector<Task> tasks
-Vector<String> projects
+UserAccount(String u, String p, String n)
-getName():String
-getUsername():String
-getPassword():String
-getTasks():Vector<Task>
-getProjects():Vector<String>
getNotifications():Vector<Notification>
-setUsername(String s):void
-setPassword(String p):void
-addTask(Task t):void
-addNotification(Notification n):void
-addProject(String name):void
-removeTask(Task t):void
-removeNotification(Notification n):void
-removeProject(String name):void
Notification
-String message
-boolean read
-int assignedDate
-int assignedTime
-UserAccount assigner
-UserAccount assignee
+Notification(String m, int ad, int at, UserAccount a1, UserAccount a2)
-getMessage():String
-getAssignedDate():int
-getAssignedTime():int
-getAssigner():UserAccount
-getAssignee():UserAccount
-setRead():void
-setAssignedDate(int i):void
-setAssignedTime(int i):void
Task
-String task
-String parentList
-boolean completed
-int assignedDate
-int assignedTime
-UserAccount assigner
-UserAccount assignee
+Task(String t, String p, int ad, int at, UserAccount a1, UserAccount a2)
-getTask():String
-getParentList():String
-getAssignedDate():int
-getAssignedTime():int
-getAssigner():UserAccount
-getAssignee():UserAccount
-setCompleted():void
-setAssignedTime(int i):void
-setAssignedDate(int i):void



Client

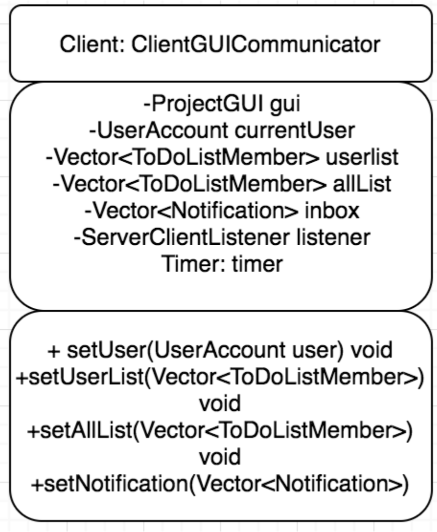
Class Dimension for Client



Interface ClientGUICommunicator

+requestLogin(Userdata) List
+addTask(Task) boolean
+deleteTask(Task) boolean
+requestList(String) List
+addList(String) boolean
+addUserToList(String, List) boolean

ClientGUICommunicator interface which will be implemented by the client class. The GUI will call these methods to signal the client that something happens in GUI.

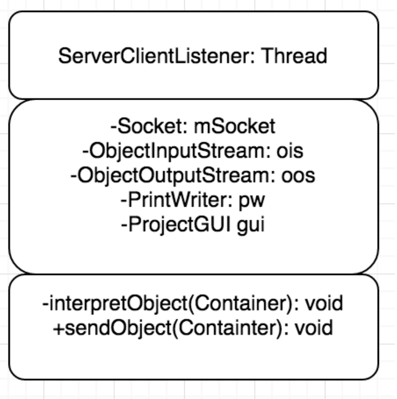


Client: ClientGUICommunicator

-ProjectGUI gui
-UserAccount currentUser
-Vector<ToDoListMember> userlist
-Vector<ToDoListMember> allList
-Vector<Notification> inbox
-ServerClientListener listener
Timer: timer

+ setUser(UserAccount user) void
+setUserList(Vector<ToDoListMember>) void
+setAllList(Vector<ToDoListMember>) void
+setNotification(Vector<Notification>)

The client class is store all the information and related components of the server

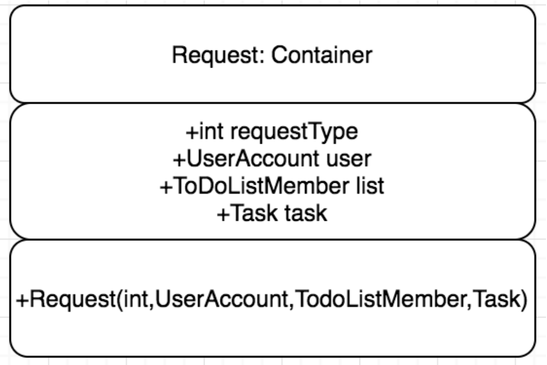


ServerClientListener: Thread

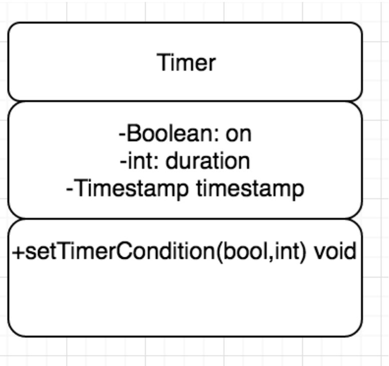
-Socket: mSocket
-ObjectInputStream: ois
-ObjectOutputStream: oos
-PrintWriter: pw
-ProjectGUI gui

-interpretObject(Container): void
+sendObject(Containter): void

ServerClientListener is the hub for transmitting the information between server and client.

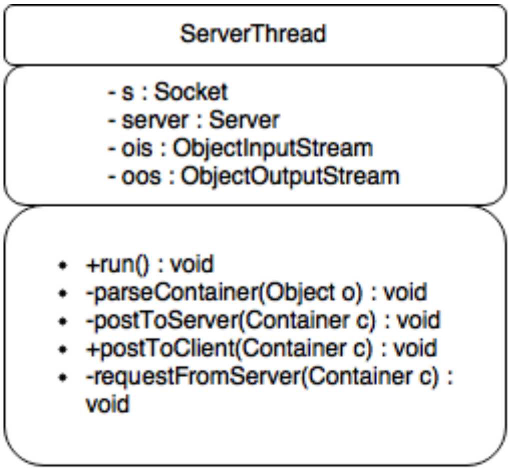


The request class is sent to the server to obtain information from the database.



Timer class is used to implement the timer function.

Server

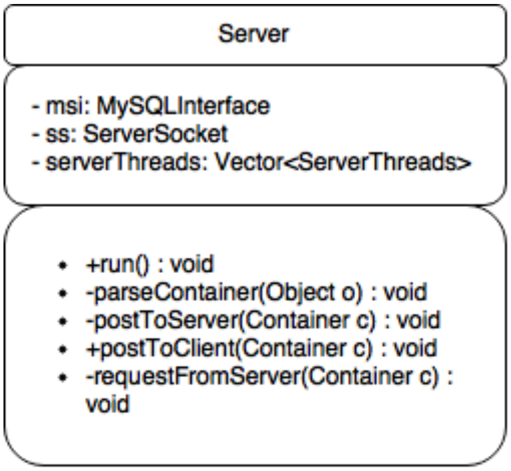


ServerThread

- s : Socket
- server : Server
- ois : ObjectInputStream
- oos : ObjectOutputStream

- +run() : void
- -parseContainer(Object o) : void
- -postToServer(Container c) : void
- +postToClient(Container c) : void
- -requestFromServer(Container c) : void

The ServerThread class contains a socket which the client is connected to. It will receive Container objects and decide whether it needs to make a request from the server, post to the server, or post to the client. The postToClient method is public so other ServerThreads may access it if notifying etc.

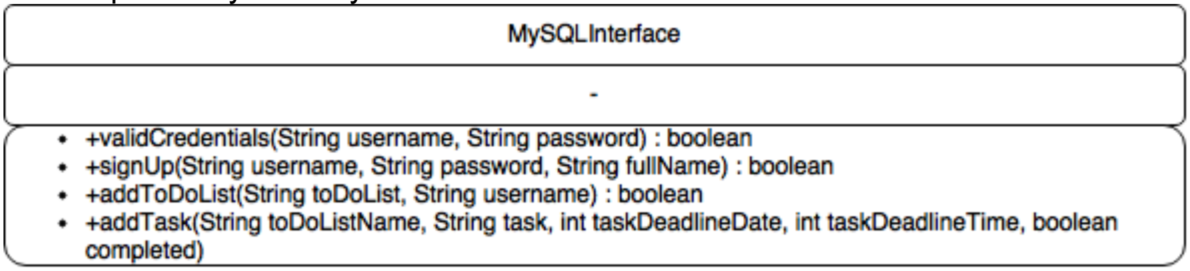


Server

- msi: MySQLInterface
- ss: ServerSocket
- serverThreads: Vector<ServerThreads>

- +run() : void
- -parseContainer(Object o) : void
- -postToServer(Container c) : void
- +postToClient(Container c) : void
- -requestFromServer(Container c) : void

The Server class has a serversocket which will accept multiple connections from clients, and pass the resulting sockets to new ServerThreads. The Server class interacts with the DB through the MySQLInterface class, which uses JDBC to make a MySQL connection. ServerThreads will pass data through server to DB, but the Server also has the unique ability to notify clients in thread

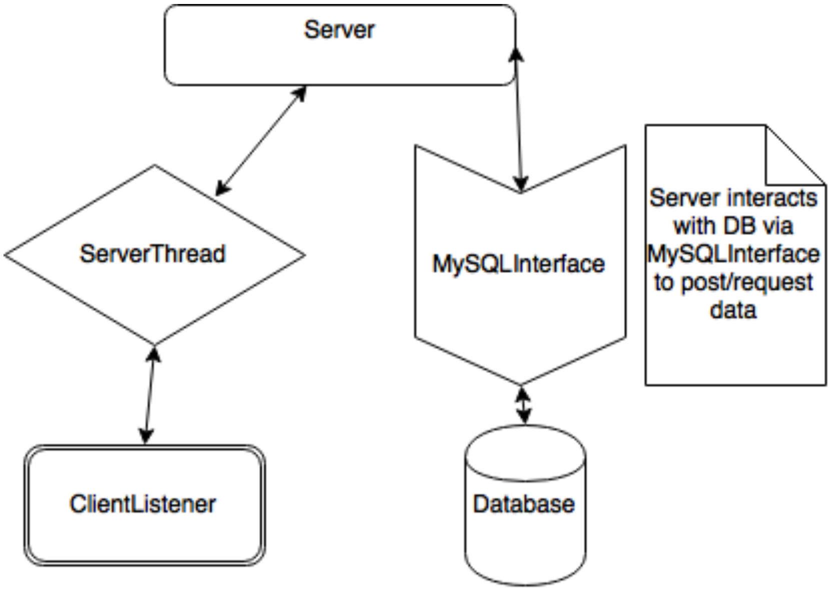


MySQLInterface

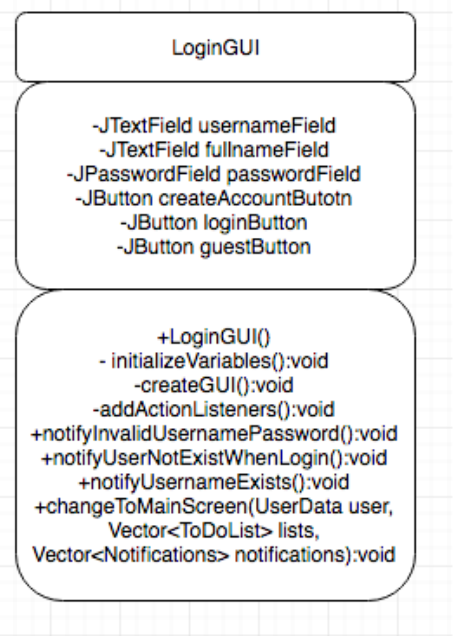
-

- +validCredentials(String username, String password) : boolean
- +signUp(String username, String password, String fullName) : boolean
- +addToDoList(String toDoList, String username) : boolean
- +addTask(String toDoListName, String task, int taskDeadlineDate, int taskDeadlineTime, boolean completed)

The MySQLInterface class uses JDBC to execute SQL commands, posting and requesting info to/from the database. The Server class will use this class to interact with the DB when clients want to post or request info


Server
ServerThread
MySQLInterface
Server interacts with DB via MySQLInterface to post/request data
ClientListener
Database

Graphical User Interface

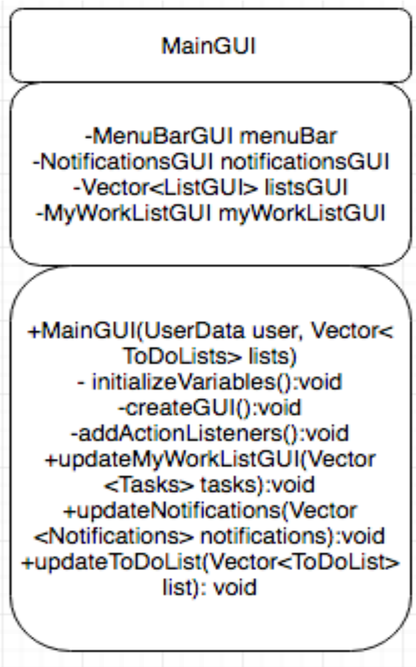


LoginGUI

-JTextField usernameField
-JTextField fullnameField
-JPasswordField passwordField
-JButton createAccountButotn
-JButton loginButton
-JButton guestButton

+LoginGUI()
- initializeVariables():void
-createGUI():void
-addActionListeners():void
+notifyInvalidUsernamePassword():void
+notifyUserNotExistWhenLogin():void
+notifyUsernameExists():void
+changeToMainScreen(UserData user,
Vector<ToDoList> lists,
Vector<Notifications> notifications):void

LoginGUI is the first window that the user sees. The user will be able to login if the user has an existing account, register with a full name if the user does not have an existing account, or login as guest. User's actions will send user inputs to the client where the client will change screens if the user input is verified or notify the user if their input is invalid.



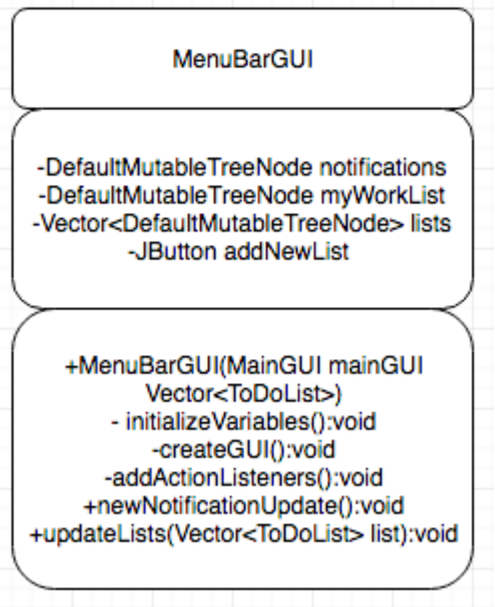
MainGUI

-MenuBarGUI menuBar
-NotificationsGUI notificationsGUI
-Vector<ListGUI> listsGUI
-MyWorkListGUI myWorkListGUI

+MainGUI(UserData user, Vector<
ToDoLists> lists)
- initializeVariables():void
-createGUI():void
-addActionListeners():void
+updateMyWorkListGUI(Vector
<Tasks> tasks):void
+updateNotifications(Vector
<Notifications> notifications):void
+updateToDoList(Vector<ToDoList>
list): void

Upon login verification, the user will be led to the MainGUI where the user will be able to select functionalities on the left side of the screen. Depending on what the user selects on the side menu bar, the user will be shown notifications tab, specified to do list tab, or

My WorkList tab which will be implemented with a CardLayout. The client will be able to update the different components of the GUI by calling the different update functions provided.

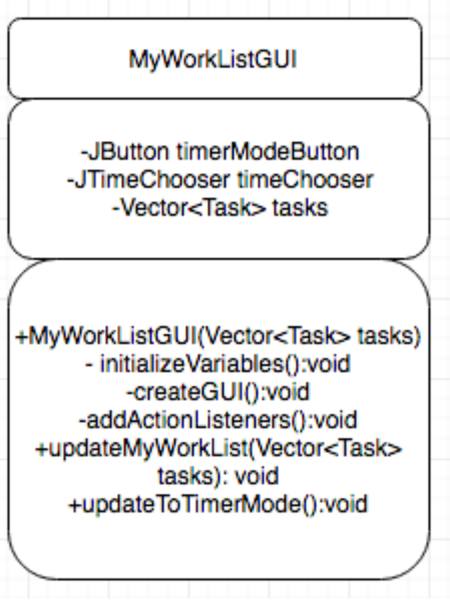


MenuBarGUI

-DefaultMutableTreeNode notifications
-DefaultMutableTreeNode myWorkList
-Vector<DefaultMutableTreeNode> lists
-JButton addNewList

+MenuBarGUI(MainGUI mainGUI
Vector<ToDoList>)
- initializeVariables():void
-createGUI():void
-addActionListeners():void
+newNotificationUpdate():void
+updateLists(Vector<ToDoList> list):void

The menu bar is implemented as a tree for the user to select functionality. MenuBarGUI class will have an instance of MainGUI so that when the user chooses a specific tab, MainGUI will be updated. In this menu bar, the user is also able to add new To-Do Lists.

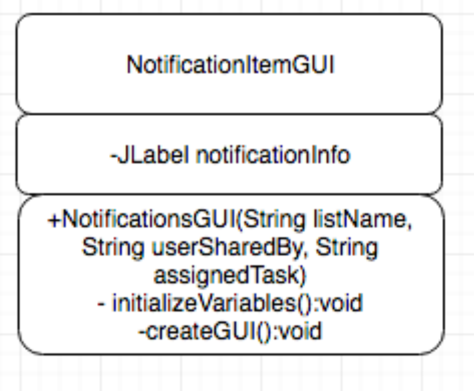


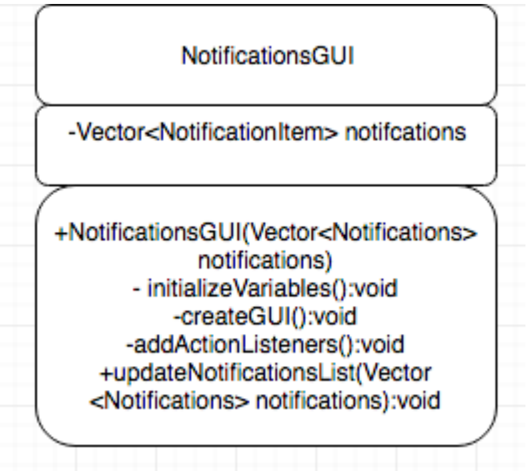
MyWorkListGUI

-JButton timerModeButton
-JTimeChooser timeChooser
-Vector<Task> tasks

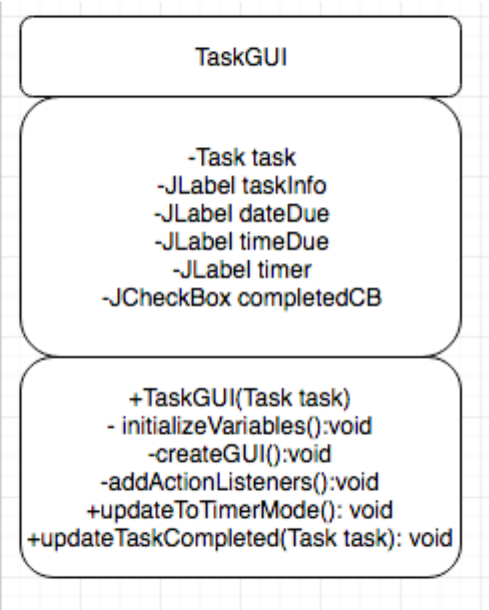
+MyWorkListGUI(Vector<Task> tasks)
- initializeVariables():void
-createGUI():void
-addActionListeners():void
+updateMyWorkList(Vector<Task>
tasks): void
+updateToTimerMode():void

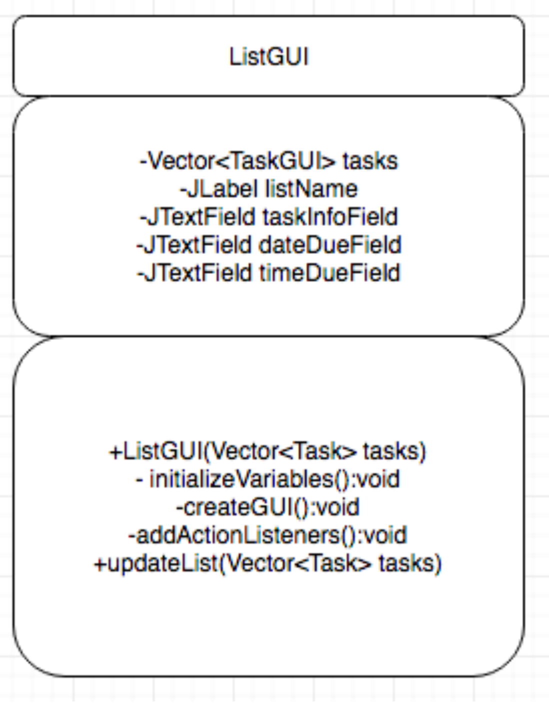
MyWorkListGUI features tasks that only pertains to the user logged in. This feature has a timer mode button where the user can select to set a timer using JTimeChooser and see tasks deadline as a countdown timer.


NotificationItemGUI
-JLabel notificationInfo
+NotificationsGUI(String listName, String userSharedBy, String assignedTask)
- initializeVariables():void
-createGUI():void


NotificationsGUI
-Vector<NotificationItem> notifcations
+NotificationsGUI(Vector<Notifications> notifications)
- initializeVariables():void
-createGUI():void
-addActionListeners():void
+updateNotificationsList(Vector <Notifications> notifications):void

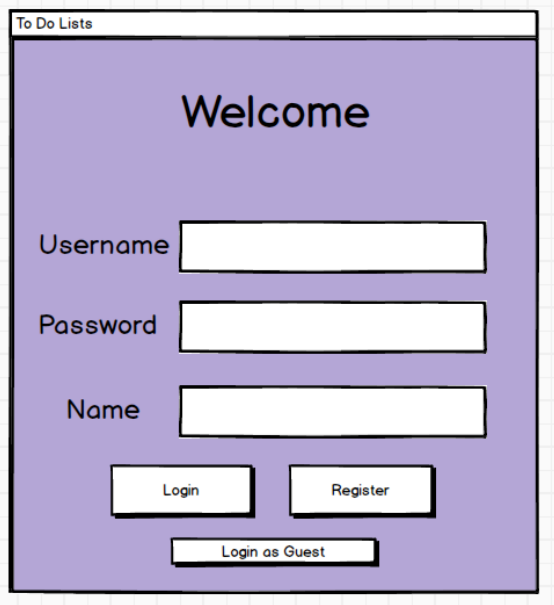
Notifications tab will show NotificationsGUI containing NotificationItemGUI objects. This panel will show notifications for the user (when someone assigns a task or shares a list). The user also has the option to clear notifications which will notify the client that the user wants to delete all notifications in his/her history.

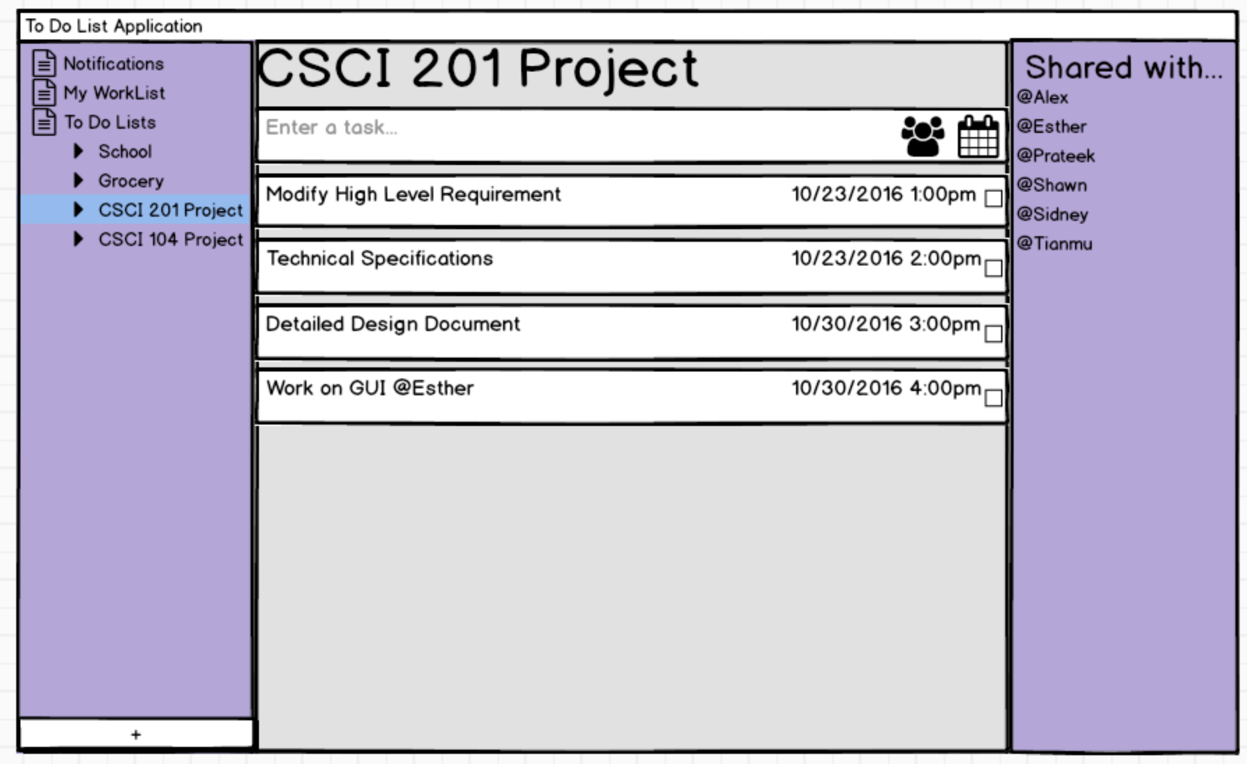



ListGUI
-Vector<TaskGUI> tasks
-JLabel listName
-JTextField taskInfoField
-JTextField dateDueField
-JTextField timeDueField
+ListGUI(Vector<Task> tasks)
- initializeVariables():void
-createGUI():void
-addActionListeners():void
+updateList(Vector<Task> tasks)

Each to do list will show a ListGUI that contains a vector of TaskGUI. ListGUI will also provide the user the option to add new tasks with a deadline. The user can tag another user using “@” character in the text field provided.

GUI Mockups



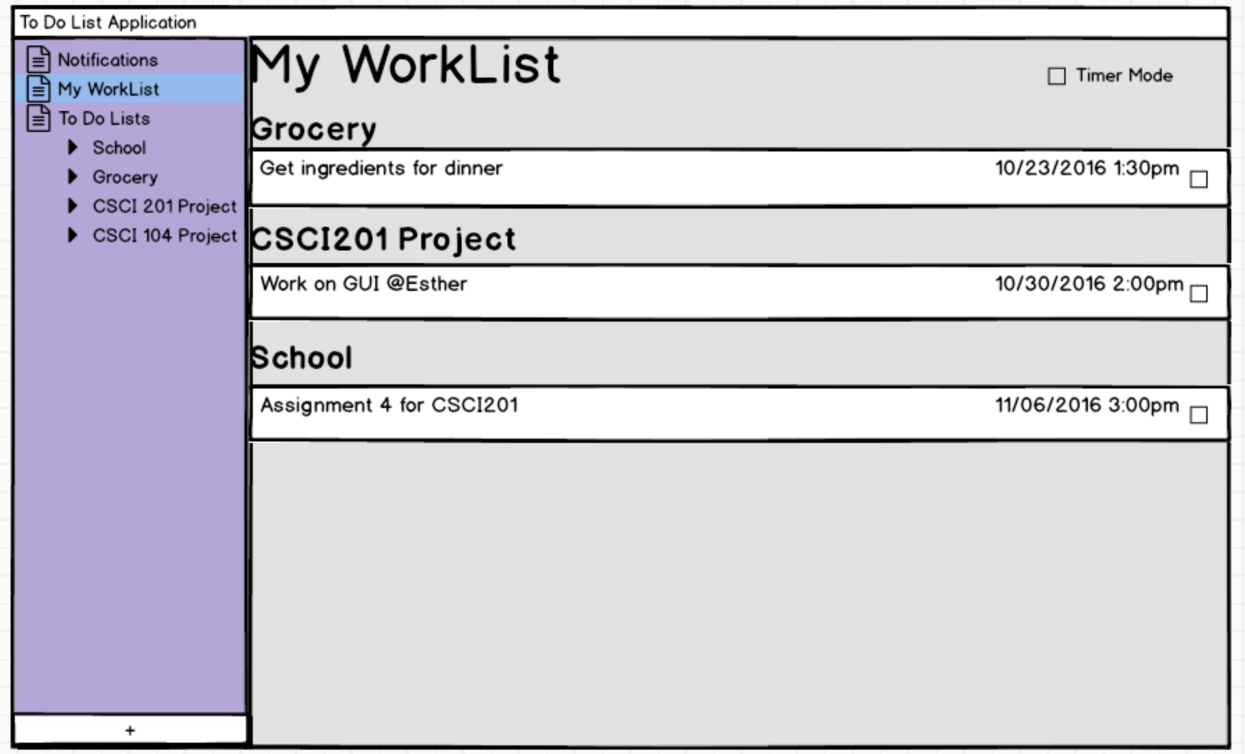


Modify High Level Requirement 10/23/2016 1:00pm □

Technical Specifications 10/23/2016 2:00pm □

Detailed Design Document 10/30/2016 3:00pm □

Work on GUI @Esther 10/30/2016 4:00pm □



My WorkList

□ Timer Mode

Grocery

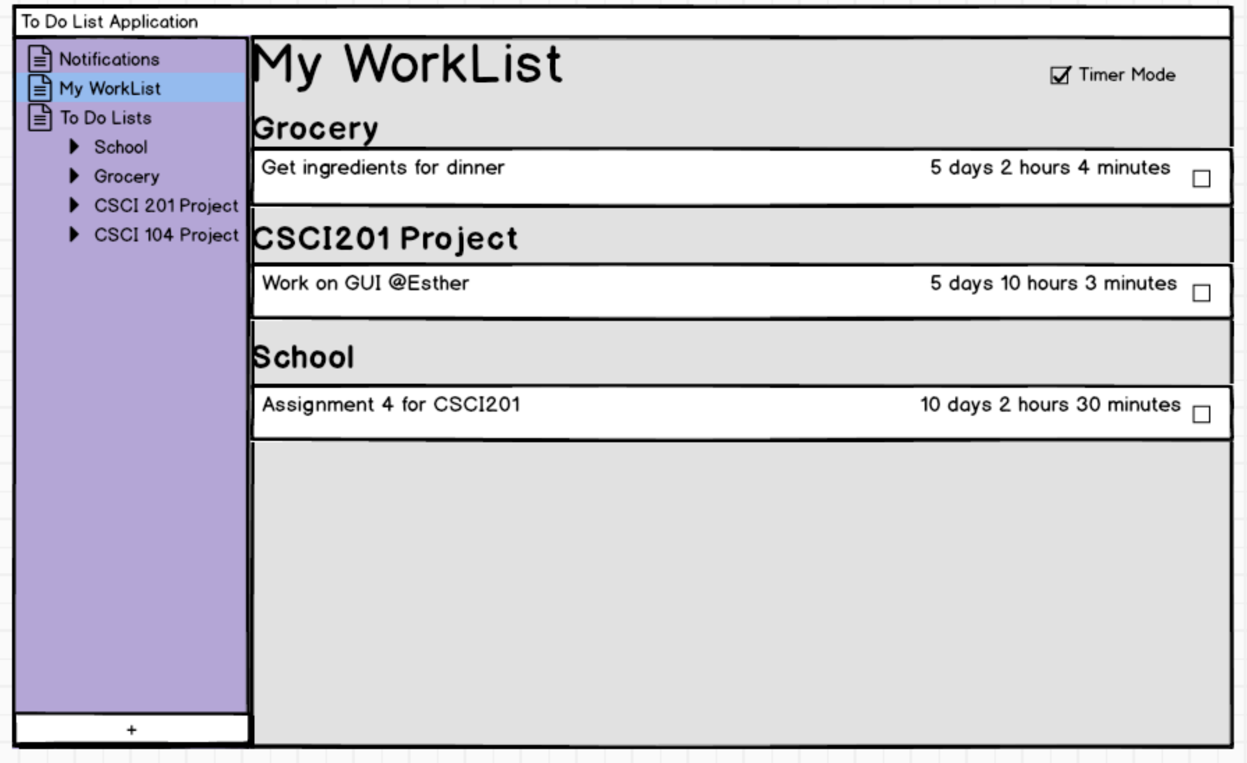
Get ingredients for dinner 10/23/2016 1:30pm □

CSCI201 Project

Work on GUI @Esther 10/30/2016 2:00pm □

School

Assignment 4 for CSCI201 11/06/2016 3:00pm □



My WorkList

☑ Timer Mode

Grocery

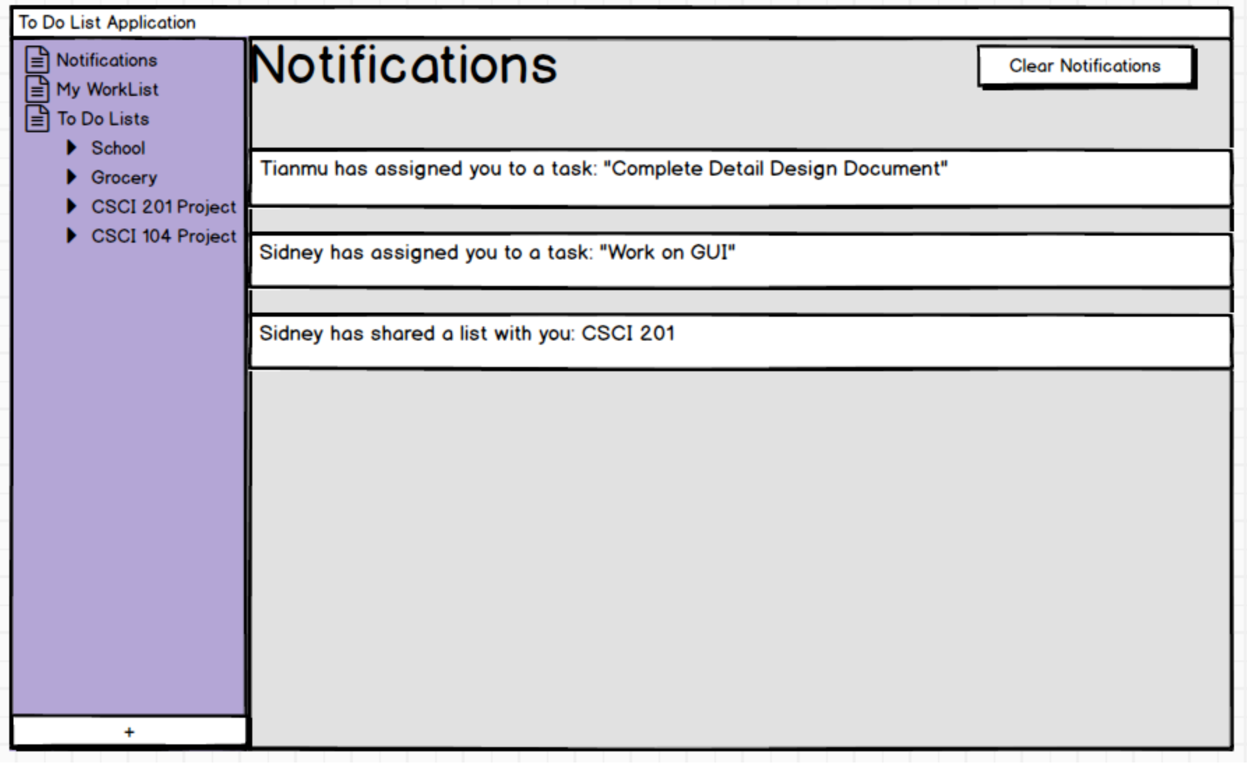
Get ingredients for dinner 5 days 2 hours 4 minutes □

CSCI201 Project

Work on GUI @Esther 5 days 10 hours 3 minutes □

School

Assignment 4 for CSCI201 10 days 2 hours 30 minutes □



Notifications

Tianmu has assigned you to a task: "Complete Detail Design Document"

Sidney has assigned you to a task: "Work on GUI"

Sidney has shared a list with you: CSCI 201

Hardware and Software requirements

Hardware
RAM: 128MB
Windows 7,8,10
Mac OS X: Intel-based Mac running Mac OS X 10.8.3+,10.9+

Software
Java 8
Eclipse IDE for Java EE Developers

Testing Document

Test #	1
Test Description	Empty login
Steps to run test	1. Leave both the username and the password field empty 2. Click “Login” 3. Enter a username, but leave the password field empty 4. Click “Login” 5. Enter a password, but leave the username field empty 6. Click “Login”
Expected Result	The Login Button should be disabled

Test #	2
Test Description	Valid login
Steps to run test	1. Enter an existing username and correct password 2. Click “Login”
Expected Result	The server will query the database for the username and password. The server will then notify the client that the username/password is a valid credential and the main To Do List application will be loaded.

Test #	3
Test Description	Invalid login - existing username, incorrect password
Steps to run test	1. Enter an existing username and an incorrect password 2. Click “Login”
Expected Result	The server will query the database for the username and password. The server will then notify the client that the username/password is an invalid credential and the error message, “Invalid login”, will be displayed.

Test #	4
Test Description	Invalid login - nonexistent username
Steps to run test	1. Enter a nonexistent username and a password 2. Click “Login”
Expected Result	The server will query the database for the username and password. The server will then notify the client that the username does not exist and the error message, “Username does not exist”, will be displayed.

Test #	5
Test Description	Empty Create Account
Steps to run test	1. Click “Create Account” 2. Leave all the fields empty 3. Click “Create Account” 4. Fill all fields, but leave the first name field empty 5. Click “Create Account” 6. Fill all fields, but leave the last name field empty 7. Click “Create Account” 8. Fill all fields, but leave the username field empty 9. Click “Create Account” 10. Fill all fields, but leave the password field empty 11. Click “Create Account”
Expected Result	The Create Account Button for all steps after the first one should be disabled

Test #	6
Test Description	Creating account - existing username
Steps to run test	1. Click “Create Account” 2. Specify first name, last name, existing username, and password 3. Click “Create Account”
Expected Result	The first “Create Account” button will lead the user to a registration panel with fields for first name, last name, username, and password. After clicking the second “Create Account” button, the server will query the database to check for existing username. The server will then notify the client that the username already exists and the error message, “Username already exists”, will be displayed.

Test #	7
Test Description	Creating account - valid registration
Steps to run test	1. Click “Create Account” on the login page 2. Specify first name, last name, nonexistent username, and password 3. Click “Create Account” on the registration page
Expected Result	The first “Create Account” button will lead the user to a registration panel with fields for first name, last name, username, and password. After clicking the second “Create Account” button, the server will update the database with the specified data, The server will then notify the client that creating account is successful and the main To Do List application will be loaded.

Test #	8
Test Description	Client populating To Do List Panel upon login verification by the server
Steps to run test	1. Login with valid credentials
Expected Result	Once the server verifies login information and notifies the client, the client populated the To Do List menu bar with different to do lists pertaining to the logged in user. The client also populated the to do list panels with tasks pertaining to the specified To Do Lists.

Test #	9
Test Description	Guest account login
Steps to run test	1. Click “login as a guest” on the login page
Expected Result	The main To Do List application will be loaded with inbox and Share ToDo List disabled.

Test #	10
Test Description	Guest account create list - invalid entry
Steps to run test	1. Login as a guest 2. Click “add list” button 3. Leave at least one field empty 4. Click “Finish” button
Expected Result	The Finish Button should be disabled

Test #	11
Test Description	Guest account create list
Steps to run test	1. Login as a guest 2. Click “add list” button 3. Fill in the title and related information 4. Click “Finish” button
Expected Result	The user should brought to the main panel after guest login. After the user clicks the “add list” button. The user will be prompted to enter the title and the related information. After the user presses “finishes”, the corresponding “To Do List” will be shown in the list on the left side. Also, the newly created list should be shown on “my list” column on the left. The task will only be stored locally. If the user logout and login again, all the data stored respect to the user will be lost.

Test #	12
Test Description	Guest account - My WorkList
Steps to run test	1. Login as a guest 2. Click on “My WorkList” on the left menu bar
Expected Result	My WorkList panel for the guest user will be loaded containing every single task that the guest user created for all to do lists. The tasks in this list will be sorted by deadline of each task and the categorized by the name of the to do list.

Test #	13
Test Description	Guest account - Notifications tab
Steps to run test	1. Login as a guest 2. Click on “Notifications” tab on the left menu bar
Expected Result	Notifications panel will be loaded for the guest user containing “Welcome” notification only.

Test #	14
Test Description	Logout functionality
Steps to run test	1. Login as an existing user or as a guest 2. Click on the Menu option at the top of the window 3. Click on menu item, “Logout”
Expected Result	The user will be led to the login screen. For the guest, all of the information entered through the application will be deleted.

Test #	15
Test Description	Creating a new To Do List - invalid entry
Steps to run test	1. Successfully login to the system 2. Click “add list” button 3. Leave at least one field empty 4. Click “Finish” button
Expected Result	The Finish Button should be disabled

Test #	16
Test Description	Creating a new To Do List - valid entry
Steps to run test	1. Successfully login to the system 2. Click “add list” button 3. Fill in the title and related information 4. Click “Finish” button

Expected Result	The user should see the main panel after login to the system. After the user clicks the “add list” button. The user will be prompted to enter the title and the related information. After the user presses “finishes”, the corresponding “To Do List” will be shown in the list on the left side. Also, the newly created list should be shown on “my list” column on the left. If the some user logout and login the system on another computer, the added To Do List still exists on the list on the left side.

Test #	17
Test Description	Assigning a task to an existing user
Steps to run test	1. Add a new task to the project 2. In the task description add a “@” that is immediately followed by the assigned user’s name in order to assign them to that task.
Expected Result	The user should receive a notification of the assigned task and have it added to their work list.

Test #	18
Test Description	Assigning a task to a nonexistent user
Steps to run test	1. Add a new task to the project 2. In the task description, add a “@” that is immediately followed by a nonexistent username
Expected Result	The client sends the server the new task information and the server informs the client that the username that the task is assigned to does not exist in the database. The user is then notified with a JDialog with a message, “The assigned user does not exist”.

Test #	19
Test Description	Adding a task to a project - unique
Steps to run test	1. Add a new task to a given project by providing a unique task description, due date, and assigned user (optional).
Expected Result	The task should be successfully added onto the project’s To-Do List

Test #	20
Test Description	Adding a task to a project - non-unique
Steps to run test	1. Add a new task to a given project and provide a task description identical to one already in the project's To-Do List (the rest of the information can be filled out as one pleases).
Expected Result	The assigning user should be notified that the task already exists in the To-Do List via a JDialog box.

Test #	21
Test Description	Adding a task with invalid deadline
Steps to run test	1. Add a new task in a given To Do List 2. Provide a task description 3. Click calendar button to open deadline setter window 4. Fill in date in text fields for deadline information that is already passed 5. Close the deadline setter window
Expected Result	The user will be notified the task deadline is invalid using JDialog and the task will not be added unless the user enters a valid deadline.

Test #	22
Test Description	Pass deadline task
Steps to run test	1. Log into the system after a task deadline has passed 2. Go to the To Do list associated with the task
Expected Result	The task with passed deadline should be shown in red

Test #	23
Test Description	Delete a task in To Do List
Steps to run test	1. Log in the system, and go to a To Do List 2. Press the "delete" button beside a task

Expected Result	The task should be deleted from the database. Also, no member sharing the To Do List is able to see this task on their GUI.

Test #	24
Test Description	Notification - adding into a To Do List
Steps to run test	1. Assign a task in a To Do List to a user for the first time 2. Check the assigned user's Notifications tab
Expected Result	The user should find a notification in its Notification tab, indicating that the user was added into a new To Do List, and a new Task has been assigned.

Test #	25
Test Description	Notification - tagged in a task
Steps to run test	1. Assign a task to a user of a To Do List 2. Check the assigned user's Notifications tab
Expected Result	The user should find a notification in its Notification tab, indicating that another user tagged you in a task in one of your to do list.

Test #	26
Test Description	Delete Notification
Steps to run test	1. Press the button "delete notification"
Expected Result	All of the notifications should be deleted from the GUI and database.

Test #	27
Test Description	Check Updated Notifications
Steps to run test	1. Press the "Notifications" tab

Expected Result	The user will be able to see all notifications stored in the database, except the notification deleted by the user previously

Test #	28
Test Description	Unread Notification Bubble
Steps to run test	1. Tag a user in a task 2. Open the Notifications tab to read about the new task
Expected Result	A red dot should be present beside the Notifications tab indicating an unread notification. One the user click on the Notifications tab, the red dot should disappear

Test #	29
Test Description	Timer mode begins
Steps to run test	1. Click on “My WorkList” 2. Click on “Timer mode” checkbox
Expected Result	The timer mode should begin and all tasks deadline information should change to a countdown timer including days and hours left on a task

Test #	30
Test Description	Timer mode on empty “My WorkList”
Steps to run test	1. Click on “My WorkList” 2. Click on “Timer mode” checkbox
Expected result	The user will be shown a message through a JDialog saying “Timer mode cannot start on an empty list”

Test #	31
Test Description	Timer mode ends

Steps to run test	1. Select duration of time period 2. Start the timer mode by clicking the “Timer Mode” button 3. Wait until time elapse
Expected Result	There is a signal in the main panel indicating that the timer mode is on. When the desired time elapses, there will be a “beep” sound alerting the user, and Timer Mode is over, and the signal disappears

Test #	32
Test Description	My WorkList View
Steps to run test	1. At the main To Do List Panel, click on “My WorkList”
Expected Results	My WorkList panel should load tasks that were either created without tagging or were assigned to the user. These tasks are categorized by specific To Do Lists, sorted by Earliest Deadline First

Test #	33
Test Description	Adding a task - no task info, tagged, deadline provided
Steps to run test	1. Tag a user in the input text field for new tasks 2. Provide a deadline to that task
Expected Results	“Add Task” button should be disabled

Test #	34
Test Description	Completing a task
Steps to run test	1. Add a new task 2. Click the checkbox next to that task.
Expected Results	The task is noted as completed and the client notifies the server of the completed task. The server then notifies all clients that the task is completed.

Test#	35
Test Description	Test Client-Server connectivity
Steps	1. Create Server 2. Create Client 3. Connect client to server
Expected Results	The connection is properly established, perhaps printing a line when socket is successfully created

Test#	36
Test Description	Test Client-Server Object transmitting
Steps	1. Send object via objectoutputstream from client to server 2. De-serialize object serverside
Expected Results	The server receives the object and is able to successfully cast to a container object

Test#	37
Test Description	Test server-DB connectivity
Steps	1. Instantiate the MySQLInterface 2. Use the server to perform the query functions of the MySQLInterface class
Expected Results	Outputs valid entries of the DB tables as per the queries

Test#	38
Test Description	Test client-DB propogation
Steps	1. Send an object from Client to Server 2. Check the DB for entries modified according to object sent to server
Expected Results	The information sent from client should be passed through to the DB. This test is a base test for many others which involve the storage of information in the DB

Test #	39
Test Description	Creating a new To-Do List - same name
Steps	1. Login 2. Click “add list” button 3. Make the name the same as one of a existing To Do List/project 4. Fill in all other necessary info as desired.
Expected Results	The list should be successfully created (a unique identifier will be made for it in the SQL database in order to distinguish it from the any other To-Do lists with the same name).

Deployment Document

To deploy this application within Eclipse, import the Final_Project_team_16.zip file into Eclipse.

This should generate a project called Final_Project_team_16 with src and resources directories.

The src directory contains the client, server, frames, database, and other packages.

Server

Go to ToDoList/src/Database/DatabaseFunctions.java, and put your MySQL Workbench

'username' and

'password' in line 11, 12

To execute the Server, run server.Main.

Client

To execute the Client, run client.Client.