## ToDo List Application Specifications Report

By: Norma Chang

January 12, 2019

Prepared for CSC 4996: Senior Project

Wayne State University

Table of Contents

**1. Requirements3**

1.1 Functional Requirements3

1.2 Non-Functional Requirements3

**2. System Architecture Diagram4**

**3. Data Flow Diagram5**

**4. Use Case Diagram6**

4.1 Use Case Diagram6

4.2 Use Case 16

4.3 Use Case 27

4.4 Use Case 37

**5. Sequence Diagram8**

5.1 Sequence 18

5.2 Sequence 29

5.3 Sequence 39

**6. Database Design10**

**7. Class Diagram11**

**8. Test Case12**

8.1 Test Case: Login12

8.2 Test Case: Add Task 12

8.3 Test Case: Delete Task13

**1. REQUIREMENTS**

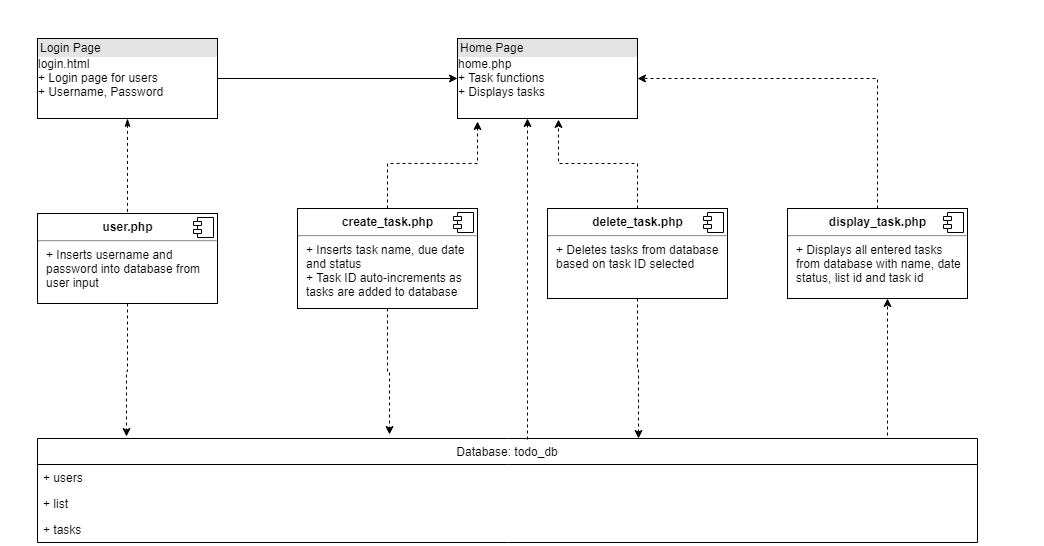
* 1. Functional Requirements

|  |  |  |
| --- | --- | --- |
| Quality Attribute | Requirement Definition | Scope/Analysis |
| Add Task | Addition of task input into the database | User must be able to add a task with its name, due date and status. |
| Delete Task | Removal of a task from the database | User must be able to delete a task that has been added to the application. |
| View Task | Display of tasks in the task list | User should be able to view all the tasks that they have added. |

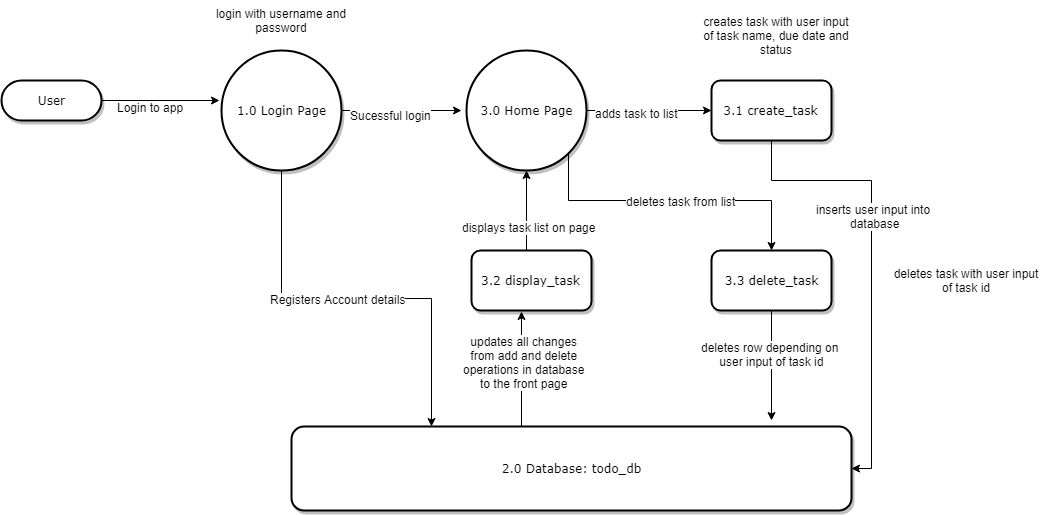
* 1. Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| Quality Attribute | Requirement Definition | Scope/Analysis |
| Scalability | Ease of adding data to the database from user input | Table size in the MySQL databse is constrained by the operating system constraints on file size. |
| Recoverability | In the event of hardware or security failures, the possibility of recovering the database and application data. | Unless proper backups are configured, in the event of a hardware/security failure, the code and the data in the database would need to be manually recreated. |
| Integrity | Consistency of the methods, values, inputs, results and data. | Avoid using methods like POST to prevent security issues like SQL injection. Limiting how the data in the database is access to prevent unauthorized modifications to the database data. |

**2. Systems Architecture Diagram**

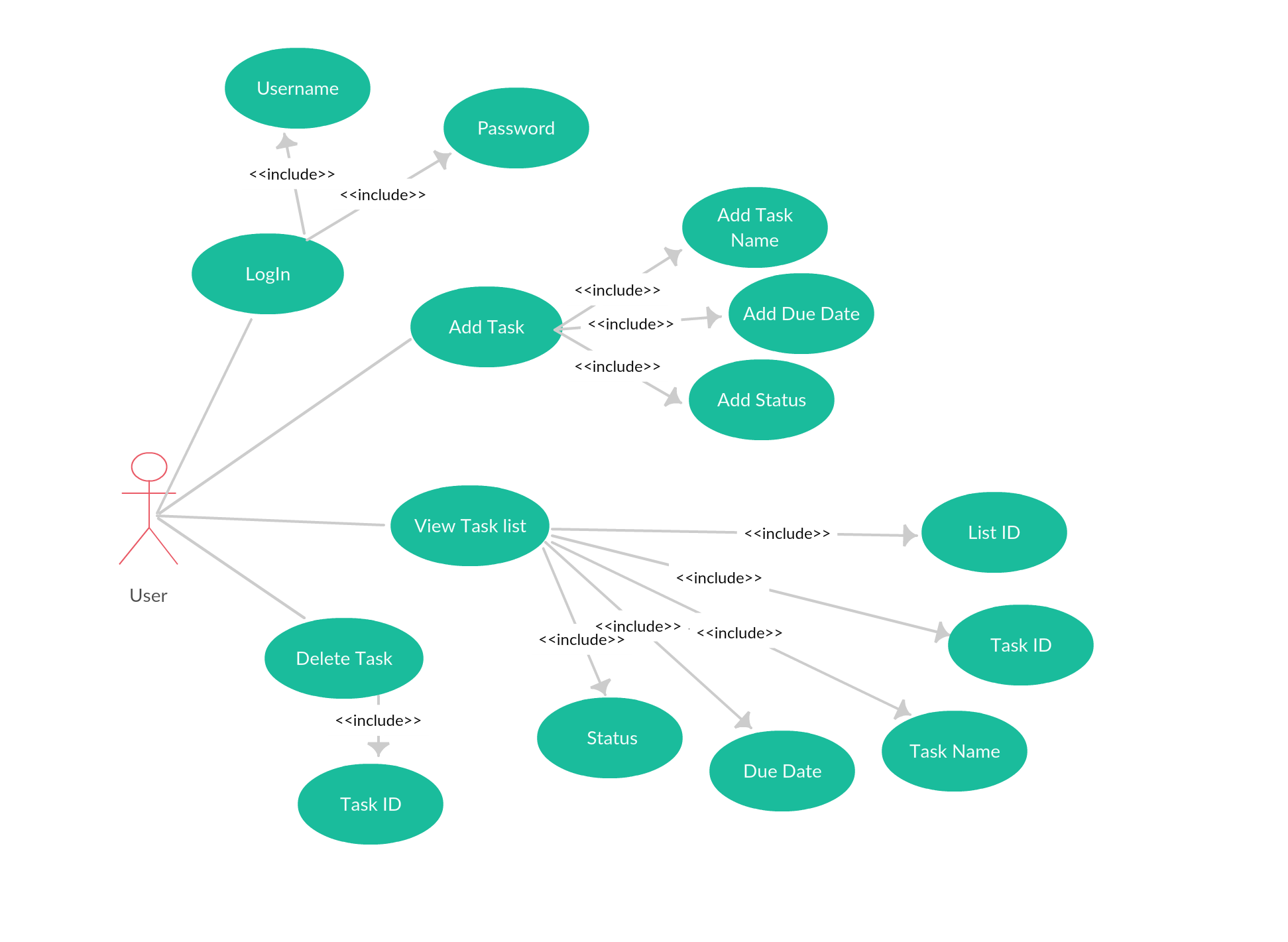


**3. Data Flow Diagram**



**4. Use Cases**

**4.1. Use Case Diagram**



**4.2. Use Case 1**

Use Case Name: Login  
Actors:

* User
* Todo list application
* Database

Description: The first page the user will encounter is the login page. The user will enter a username and password into the fields and click submit. Once submitted, the applications will say registering account if it is a new account or logging in if it is an existing account.

**4.3. Use Case 2**

Use Case Name: Add Task

Actors:

* User
* Todo list application
* Database

Description: After the user provides the name of the task, due date and status, then they can click the submit button to add the task. The system will acknowledge the addition and update the todo list view on the homepage. The user will also be able to continue adding tasks.

**4.4. Use Case 3**

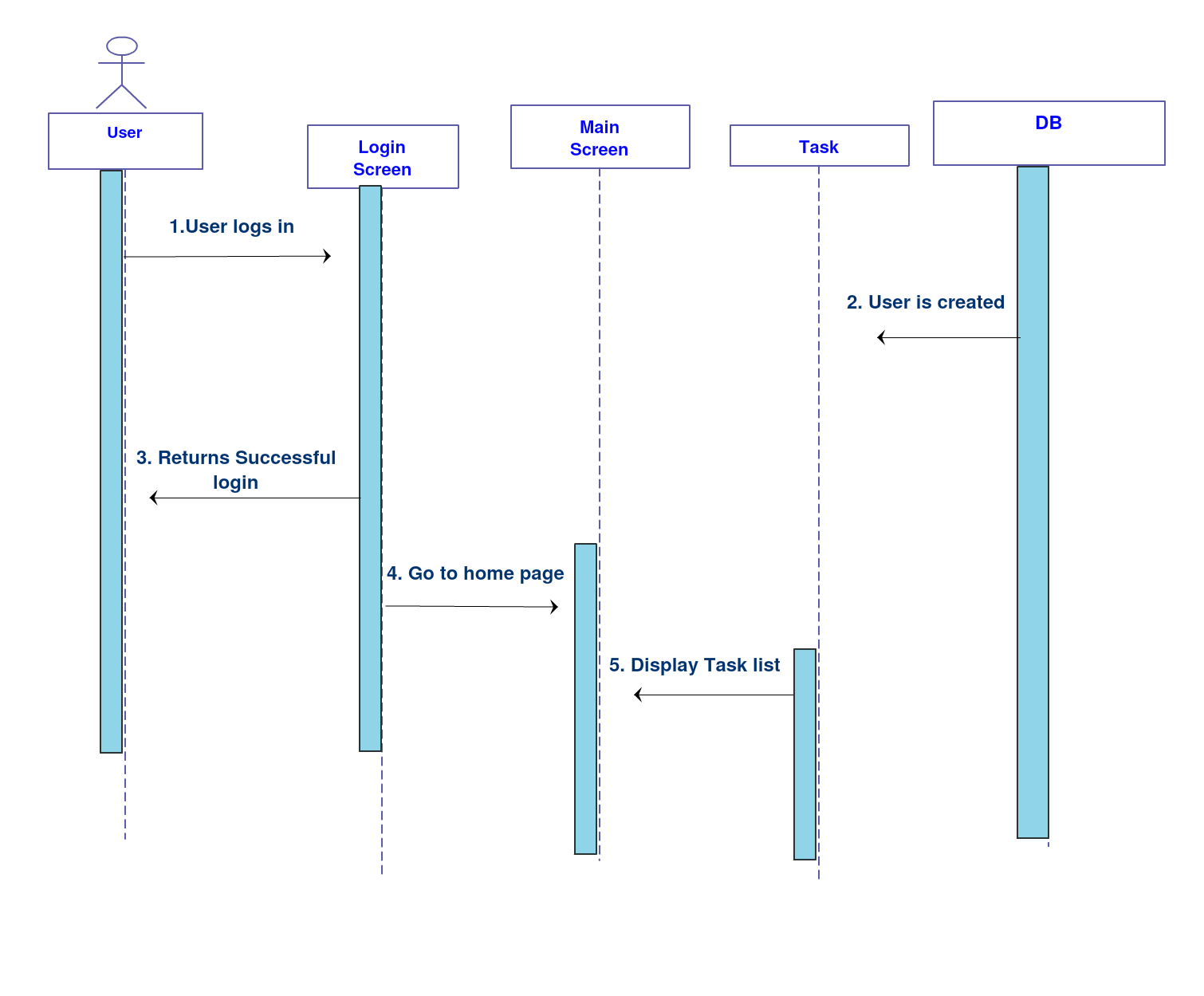
Use Case Name: Delete Task  
Actors:

* User
* Todo list application
* Database

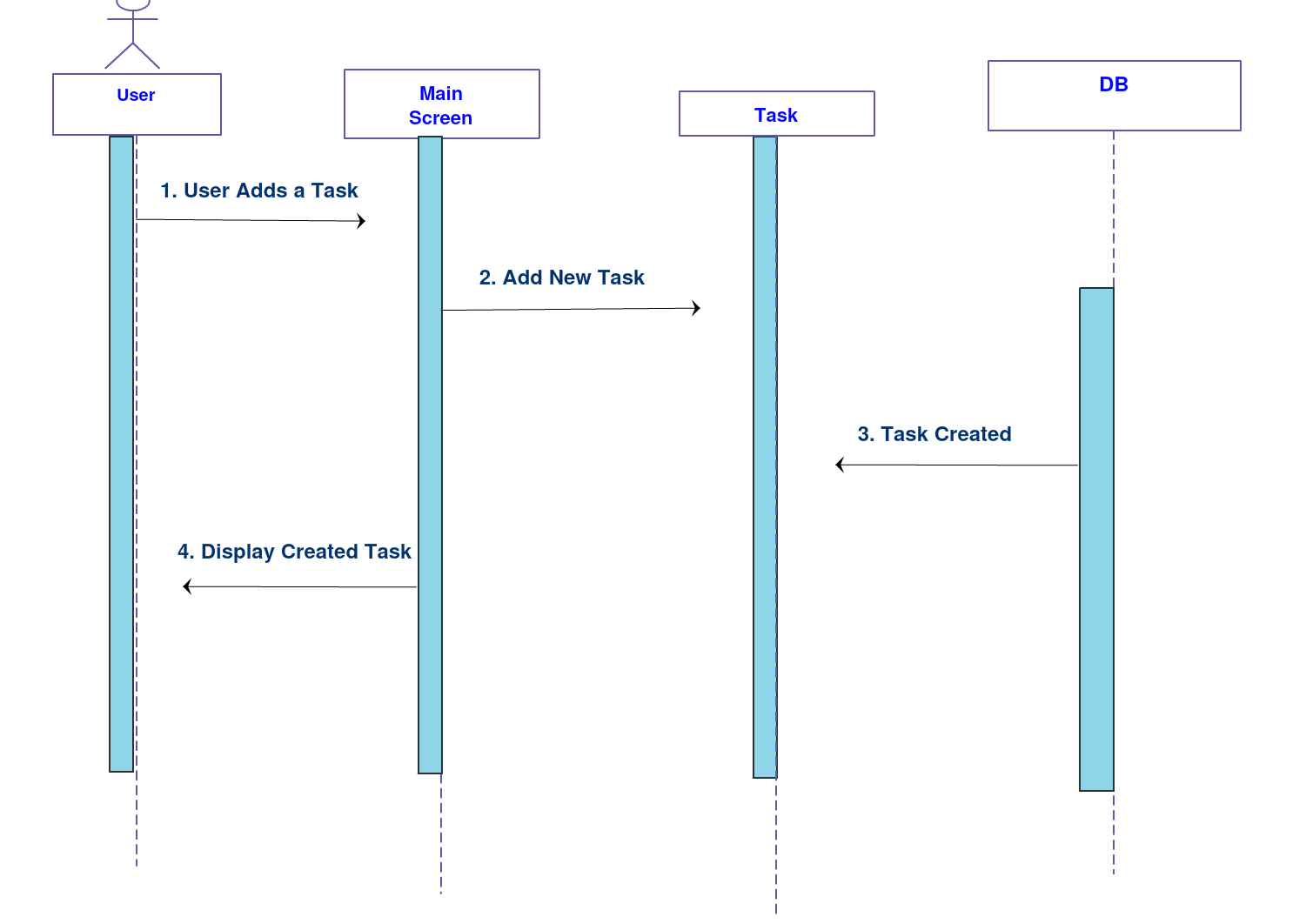
Description: On the homepage, the user will be able to see the task id of an task that they would like to delete. The user will enter the task id into the Task ID field and click the delete button. The application will then acknowledge the deletion and redirect back to the home page where the todo list is updated to reflect the deletion.

**5. Sequence Diagram**

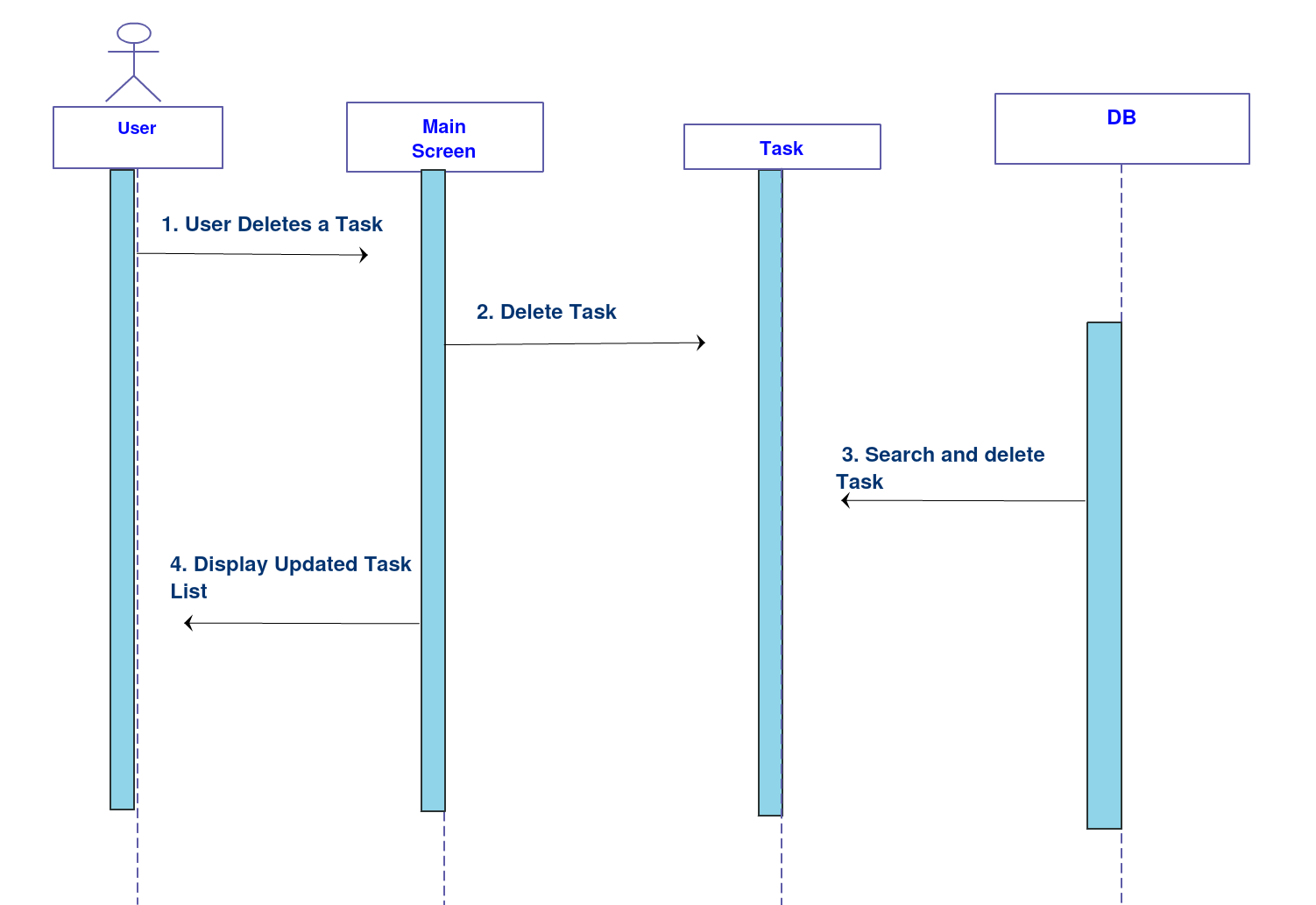
**5.1: Sequence 1 - Login**



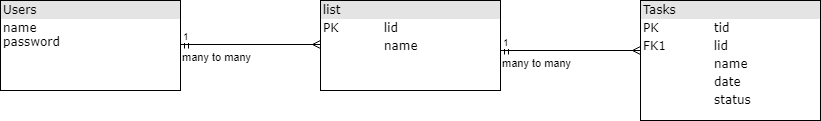
**5.2: Sequence 2 – Add Task**



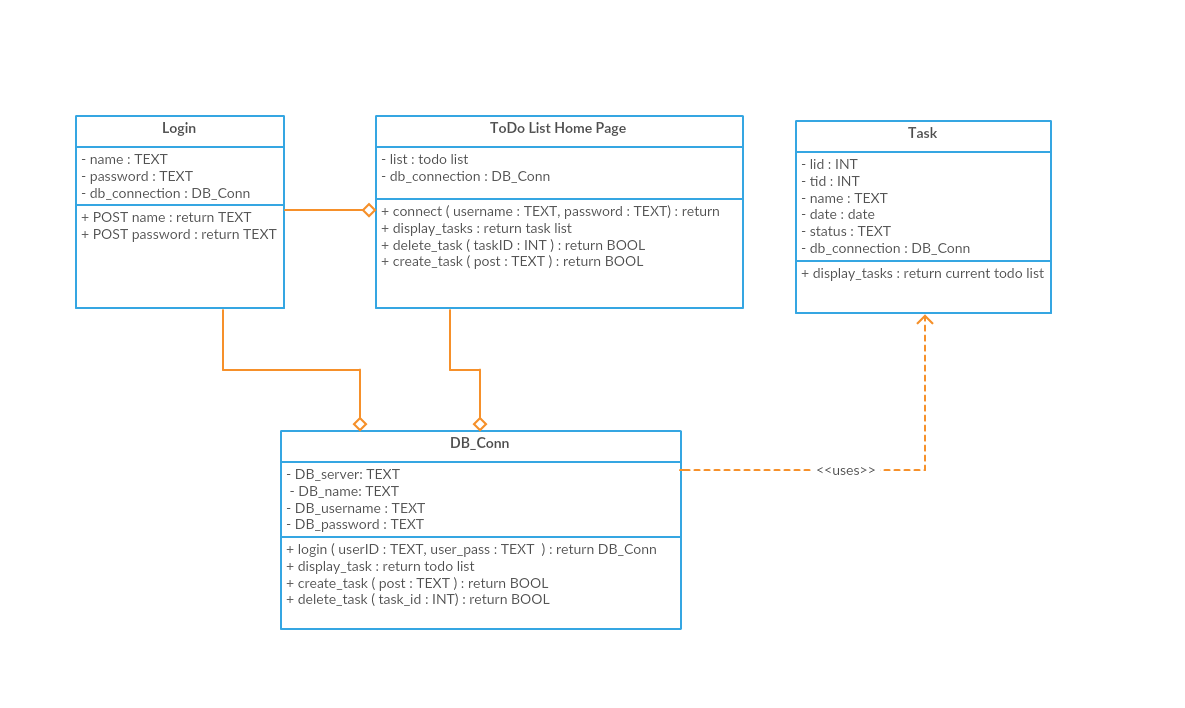
**5.3: Sequence 3 – Delete Task**



**6. Database Design Diagram**



**7. Class Diagram**



**8. Test Case**

**8.1. Login**

8.1.1 Login - Successful

|  |  |
| --- | --- |
| Test Case Name | Login – Login Successful |
| Test Case ID | TC-1 |
| Preconditions | User account from registration |
| Postconditions | User can log in and view Todo List Dashboard |
| Test Steps | 1. Open Application  2. Enter Username and Password (account will be automatically registered if it does not exist)  3. Click Login |
| Expected Results | Application displays Todo List Home Page |

**8.2. Add Task**

8.2.1 Add Task - Due

|  |  |
| --- | --- |
| Test Case Name | Add Task – Add Task Due |
| Test Case ID | TC-2 |
| Preconditions | Successful completion of login |
| Postconditions | Task is added to the database and displayed in the Todo List view |
| Test Steps | 1. Under Add Task, Enter “Due Task” in the Task Name field 2. Click the drop down in the calendar field and select “01/14/2019” 3. Type “Due” in the Task Status field 4. Click Add Task |
| Expected Results | “Due Task” will display under Todo List |

8.2.2 Add Task - Started

|  |  |
| --- | --- |
| Test Case Name | Add Task – Add Task Started |
| Test Case ID | TC-3 |
| Preconditions | Successful completion of login |
| Postconditions | Task is added to the database and displayed in the Todo List view |
| Test Steps | 1. Under Add Task, Enter “Started Task” in the Task Name field 2. Click the drop down in the calendar field and select “01/10/2019” 3. Select “Started” in the Task Status field 4. Click Add Task |
| Expected Results | “Started Task” will display under Todo List |

**8.3. Delete Task**

8.3.1 Delete Task - Started

|  |  |
| --- | --- |
| Test Case Name | Delete Task – Delete Task Started |
| Test Case ID | TC-4 |
| Preconditions | Successful completion of login and TC-2 |
| Postconditions | Task is deleted from database and list view will be updated |
| Test Steps | 1. Under Delete Task, enter the task id of the “Started Task” 2. Click Delete |
| Expected Results | “Started Task” will be removed under the task list |