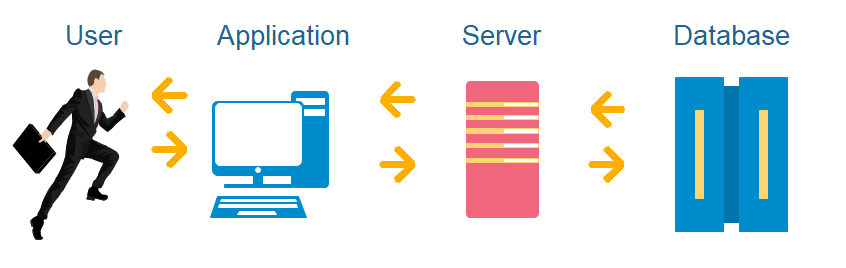
Brandon Jackson fa4269

Functional Requirements:

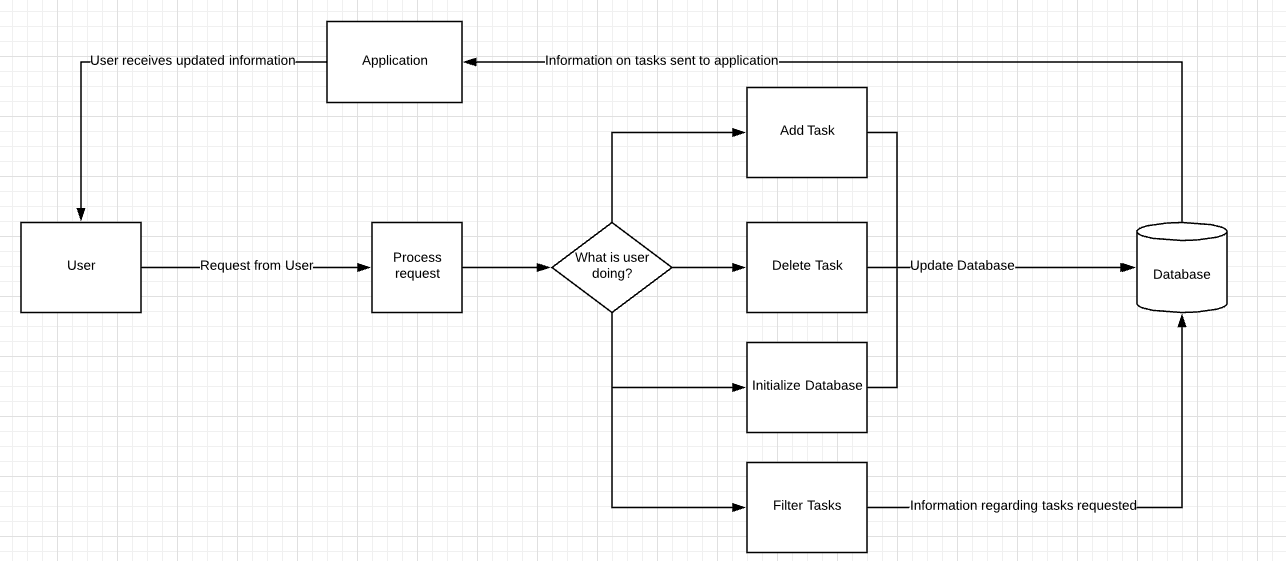
1. The user can add tasks with name, date, and status as desired
2. The user can remove tasks with id and status of their choice
3. The user can see all tasks on screen at once
4. The user can filter tasks by status
5. The system clears out all current tasks and ensures database is structured properly when initialized
6. Total tasks in the system, and tasks in each status are counted and displayed for user

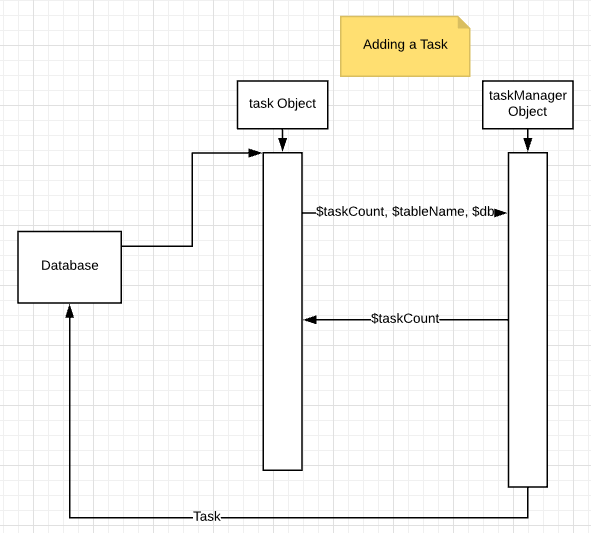
Nonfunctional Requirements:

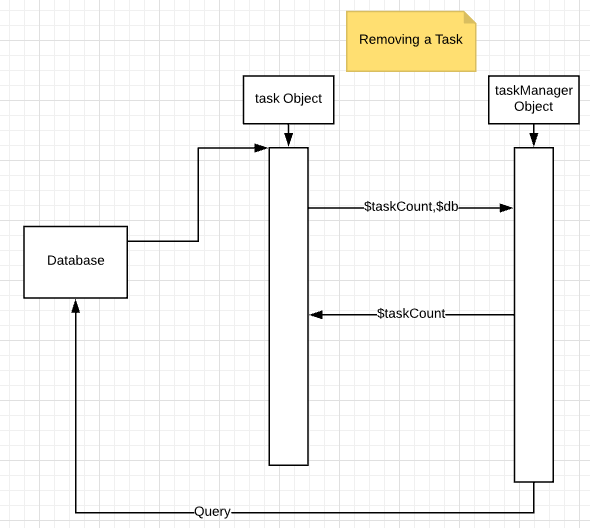
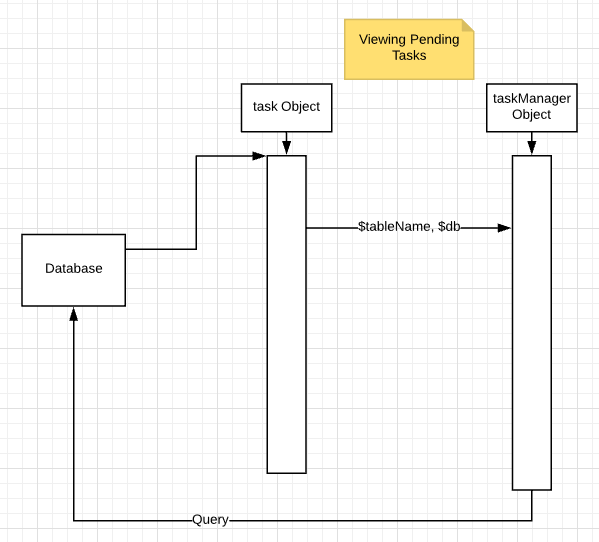
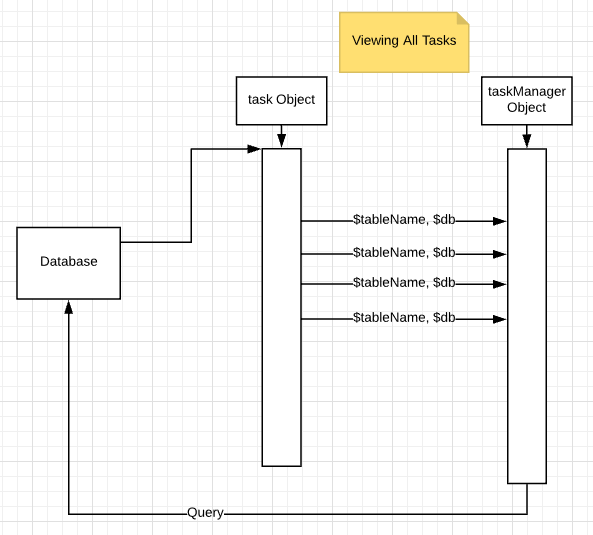
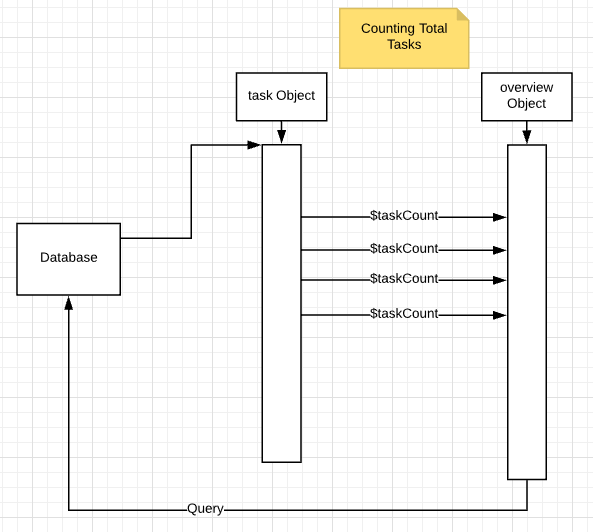
1. The program operates entirely on one webpage, never loading a different one
2. The database can be tested using the initialization button if needed
3. The database is manageable entirely through the program
4. The database can hold as many tasks as the user needs, not having a cap
5. When the user requests information it should be displayed in under 5 seconds
6. If for some reason the database becomes unstable, it can be reset easily

System Architecture Diagram

Dataflow Diagram

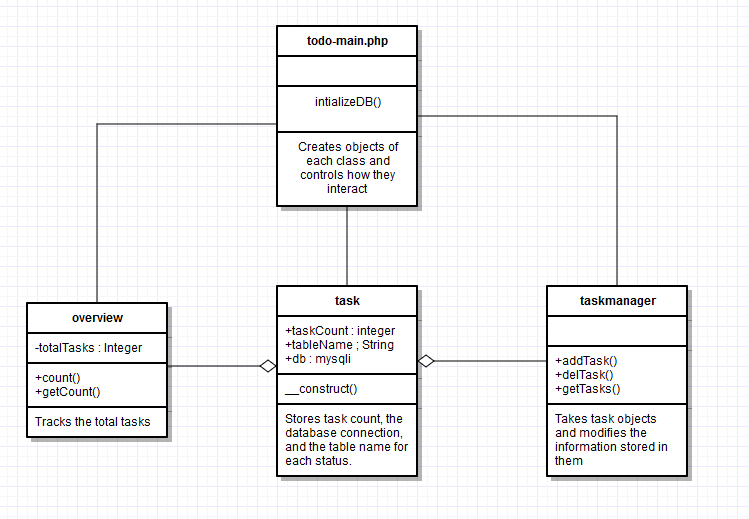


Sequence Diagrams

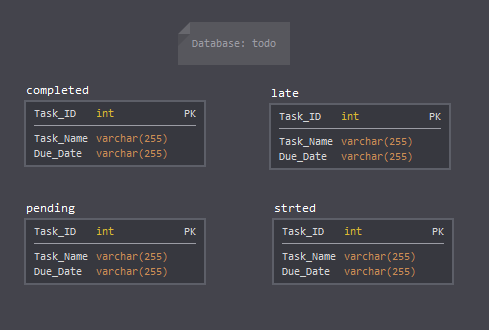


Use Cases:

1. Primary Actor: User
   1. Scenario: Adding Task
      1. Navigate to application page (localhost/todo-main.php)
      2. Initialize Database (Optional if not first time)
      3. Fill in forms for Task Name, Task Date, Status, and click ‘Submit Query’
      4. Click on desired filter to view task was added
2. Primary Actor: User
   1. Scenario: Deleting Task
      1. Navigate to application page
      2. Initialize Database
      3. Click ‘Total Tasks’ to view all tasks
      4. Enter Task ID and Status of task to remove and click ‘Submit Query
3. Primary Actor: User
   1. Scenario: Resetting Todo List
      1. Navigate to application page
      2. Press Initialize Database remove all current entries
4. Primary Actor: User
   1. Scenario: Checking Pending Tasks
      1. Navigate to application page
      2. Filter tasks by pressing ‘pending tasks’ button
5. Primary Actor: User
   1. Scenario: Changing a task status
      1. Navigate to application page
      2. Filter task and get id of task to update
      3. Delete the old task by entering ID and Status
      4. Add the task back with old info and new status
6. Primary Actor: User
   1. Scenario: Checking all tasks
      1. Navigate to application page
      2. View all tasks by pressing ‘Total Tasks’

Class Diagram

Database Diagram



Test Cases

1. Summary: Adding a node
   1. Prerequisites – Database is initialized
   2. Procedure – Enter data into Add Task field and submit
   3. Test Data – Task Name: Go to store Task Date: 1/15 Status: pending
   4. Expected – Task was added to database, Total tasks incremented
   5. Results – Task was added to database, Total tasks incremented
2. Summary: Deleting a node
   1. Prerequisites – Database is initialized, at least 1 task in database
   2. Procedure – Enter data into Delete Task field and submit
   3. Test Data – Task ID: 1 Status: pending
   4. Expected – Task was removed, Total tasks decremented
   5. Results – Task was removed, , Total tasks decremented
3. Summary: Filtering by pending tasks
   1. Prerequisites – Database is initlaized, at least 1 pending task in database
   2. Procedure – Press pending tasks button
   3. Test Data – Pending Tasks: 3
   4. Expected – Pending tasks displayed on screen
   5. Results – Pending tasks displayed on screen
4. Summary: View all tasks
   1. Prerequisites – Database Initialized, at least 1 task in database
   2. Procedure – Press Total Tasks button
   3. Test Data – Completed Tasks: 3, Started Tasks: 1, Pending Tasks: 0, Late Task: 5
   4. Expected - All tasks displayed on screen
   5. Results – All tasks displayed on screen
5. Summary: Resetting database, removing all tasks
   1. Prerequisites – Database initialized
   2. Procedure – Press Initialize Database button
   3. Test Data – Completed Tasks: 2, Started Tasks 2
   4. Expected – All tasks removed from database
   5. Results – All tasks removed from database
6. Summary: Adding a duplicate task
   1. Prerequisites – Database initialized, at least 1 task in the database
   2. Procedure – Add a task using the same parameters twice in a row
   3. Test data – Task Name: Go to store, Task Date: 1/15, Status: Pending
   4. Expected – Task will only be added in once
   5. Results – Task appears twice in a database with different ID’s

Work Breakdown:

Task: Learning basics of new technologies and languages

Estimated No of hours: 4

Actual time Spent: 5

Notes: Hard to learn php and html, wasn’t too hard once I got the basics down but not knowing them made the total time of the project go up A LOT

Task: Add Task

Estimated No of Hours: 3

Actual time spent: 4

Notes: getting information from the browser proved to be a lot more difficult than I had anticipated but once I figured it out it was rather simple from there

Task: Delete Task

Estimated No of Hours: 3

Actual time spent: 1

Notes: I reused the same basic principles from ‘Add Task’ so it wasn’t too hard to figure out

Task: Filter tasks by status

Estimated no of hours: 3

Actual time spent 5

Notes: figuring out how to display the results of the query on the screen proved to be quite difficult and I spent a lot of time on this problem

Task: Count tasks

Estimated no of hours 1

Actual time spent: less than 1 hour

Notes: Simple once I changed around my classes