Part 1: Fixing Issues from HW2

In HW2, you played the role of a *software consultant* hired to identify the security vulnerabilities in a Web application. In this homework, you will now play the role of the *software developer* that will fix the identified security vulnerabilities.

- **Q1:** Change the code provided to fix ~~all~~ ***at least two*** of the vulnerabilities you found in the web application.
- **Q2:** Include in the code comments with an explanation about the fix.

```python
# Deletes a task (based on its primary key) and redirect it back to index page
def delete(request, pk):
    ### CHANGE #1 ###
    # Added authentication check to prevent deleting tasks without logging in
    if request.user.is_authenticated:
        # uses ORM to delete the task
        task = Task.objects.get(id = pk)
        task.delete()
        # redirects user to index page
        return HttpResponseRedirect(reverse(f'tasktracker:index'))
    else:
        return HttpResponseRedirect(reverse(f'login'))
```

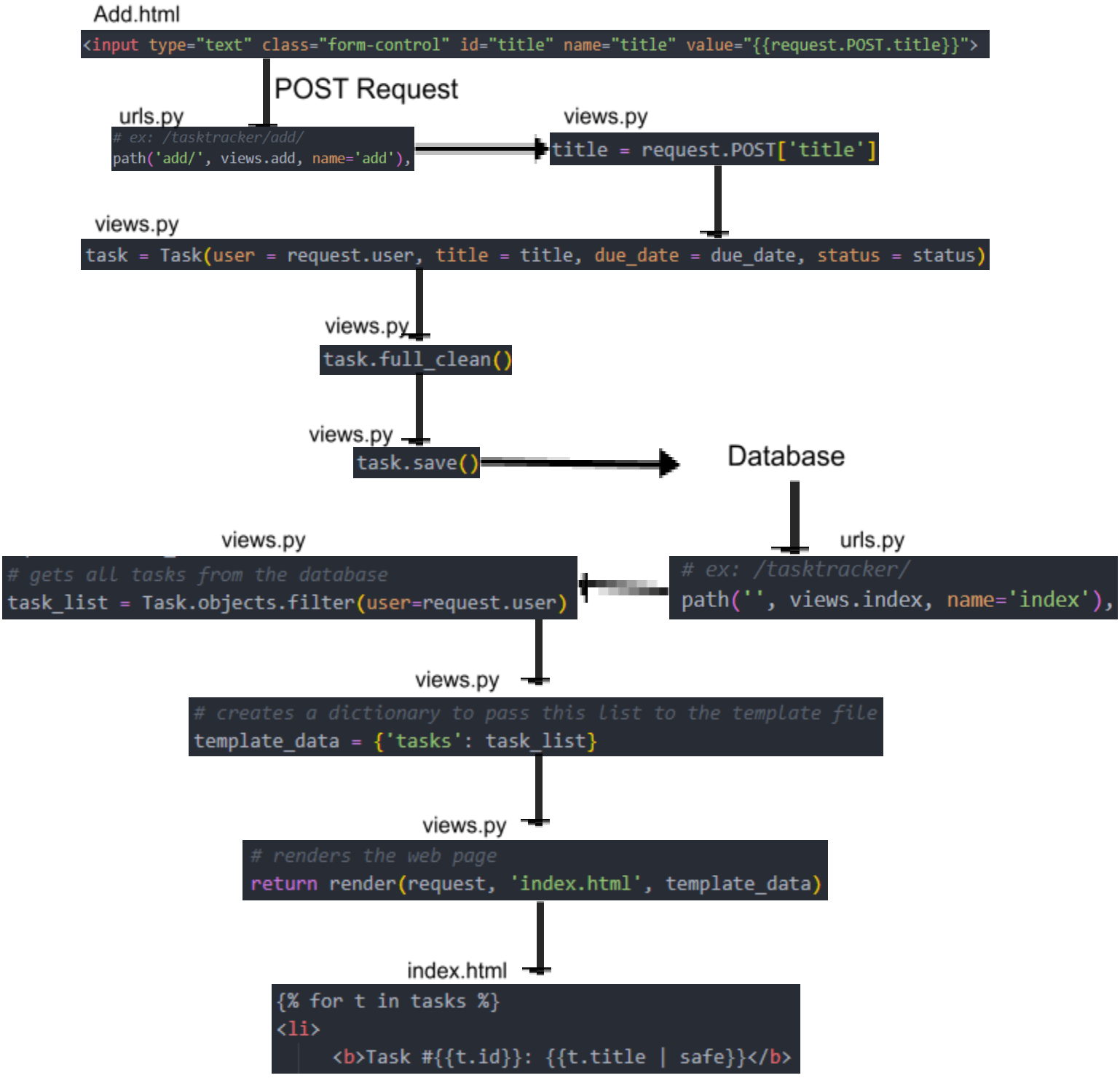
```python
# does input validation (full_clean() throws an exception if validation fails)
try:
    task.full_clean()
    # if no exception was thrown, form was validated
    # we proceed to save the task in the database
    # with connection.cursor() as cursor:
    #   cursor.executescript(f"INSERT INTO tasktracker_task(user_id, status, due_date, title) VALUES ({request.user.id},'{status}', '{due_date}', '{title}')")
    #
    ### Change #2 ###
    # Changed the raw SQL command to be task.save() to prevent SQL injection
    task.save()
except ValidationError as e:
    # renders the web page again with an error message
    return render(request, 'add.html', {"errors": e.message_dict})
```

Part 2: Identification of Threats

Q1) Consider that we have the (simplified) list of entry points, trust levels, and assets above for the application provided in the HW2 as shown in the tables above. Write down 1 threat for this application using the threat model profile shown in class:

ID	1
Name	Improper Authorization
Description	An adversary may try to tamper with the data in the database because there was previously no authorization on the delete page. They could insert “tasktracker/delete/8” into the URL, which would navigate to the delete view and delete task number 8. This could all be done from the login page without authentication.
STRIDE Classification	- Tampering - Elevation of Privilege
Mitigated?	Yes
Known Mitigations	Proper authentication on the delete task page
Entry Points	1.4: Delete Task page
Assets	1.2: Task Details

Q2) Create a data flow diagram for ***one feature of your choice*** of the HW2's application.



Part 3: Research Synthesis

Graduate students are expected to have evidence of *research synthesis* during the semester. To fulfill this requirement, please read the following paper and answer the questions below:
Mai, Phu X., et al. "Modeling security and privacy requirements: a use case-driven approach." Information and Software Technology 100 (2018): 165-182.

1) **Summarize the lessons learned while the authors applied their approach to an industrial healthcare project and from the interviews with engineers.**

This study applied their new approach (Restricted Misuse Case Modeling or RMCM) to a real-world healthcare project. The researchers learned many lessons while implementing this approach and interviewing engineers. The interviewed engineers reported that RMCM helped make security and privacy requirements more precise and complete. By structuring misuse cases and mitigation scenarios with templates, the approach reduced ambiguity that typically arises when using informal descriptions. Threats were clearly linked to functional use cases, and alternate threat flows were easier to model and understand. This structure allowed teams to capture essential information such as actors, preconditions, assets, and possible attack steps systematically, which facilitated clearer communication among team members and stakeholders. One major benefit was the traceability between misuse cases, use cases, and mitigation strategies. Engineers can determine which security and privacy threats affected certain functional requirements and how particular mitigation strategies addressed multiple threats. This explicit mapping helped ensure full coverage and enable the reuse of mitigation measures across different scenarios to reduce the duplication of effort. Additionally, the structured templates and restricted language used in RMCM supported automated consistency checks. The interviewed engineers noted that inconsistencies between diagrams and textual specifications, or missing mitigation strategies for certain threats, were easier to detect. This early detection of gaps in the requirements improved the quality and reliability of the security and privacy modeling process. RMCM has a slight learning curve as the engineers noted that mastering the use of the templates, diagrams, and keywords takes time. Despite the upfront effort, engineers found that the additional capabilities gained were worth the effort. The engineers also noted that the phrasing required for the threat scenarios was very specific and often felt limiting. However, upon using the tool more, they eventually appreciated the more structured descriptions, which can allow for automation in the future. Overall, the approach proved practically useful in a multi-device healthcare environment, where data privacy and security are critical. By integrating threat modeling into the familiar use case framework, RMCM aligned with existing development practices, which eased its adoption and improved the

relevance of security and privacy analysis in the system design. Engineers indicated through the interviews that they would benefit from further tool support, such as automated generation of security test cases. The current tool showed promise, but there is plenty of room for growth in the future with additional tooling to reduce manual effort.