CS-2450-002 | 2025 Spring

TaskZen

A Task Manager that keeps you on track and organized.

Monday, April 21, 2025

MEET THE TEAM

Group: 4

Devin Winters

Jacob West

Vlad Kashchuk

Brippney Vargas

Keomony Mary

What is

TaskZen

TaskZen is a simple, easy-to-use web-based task management application designed to help students to stay organized and on top of their responsibilities. Whether you're handling assignments, projects, exams, or personal tasks - TaskZen can help you organize your schedule to your liking.

Table Of Contents

Planning & Strategy

- Sprints
- Issue Assignment and Prioritization
- Role Switching
- Team Meetings

Analysis

- High-Level Analysis
- Low-Level Analysis

Development

- Codebase Setup
- Requirement Assessment
- Feature Implementation
- Refactoring and Optimization

UML

- Class Diagram
- Sequence Diagram
- Use Case Diagram

Wireframe

- Initial Wireframe
- Final UI Layout

Features

- CRUD for Tasks
- Dark/Light Theme
- Task Pie Chart
- User account management system

Testing

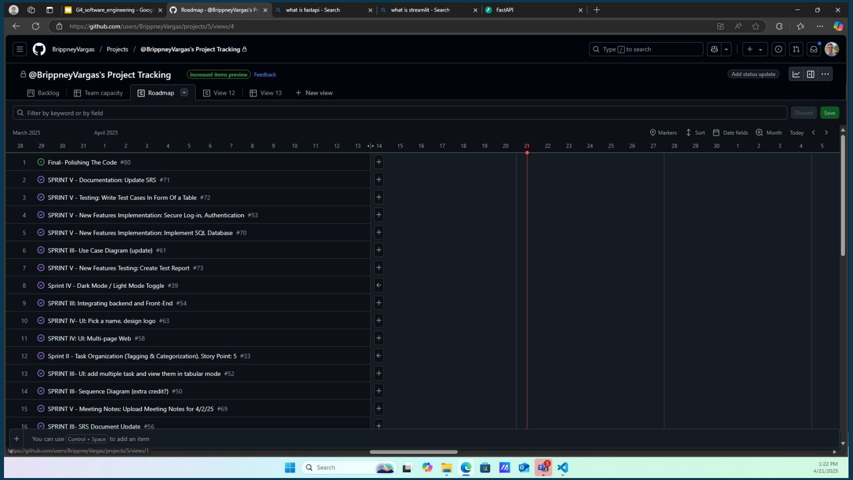
- Unittesting
- Integration Testing
- End-to-End
- Manual UI Testing

Planning & Strategy

Planning

Weekly, sometimes biweekly, meetings were held to decide on task assignments and general design questions. Online meetings were sometimes held to allow for pair programming and group debugging.

The Product Owner was in charge of creating issues/product backlog on the GitHub Project page.



G4_software_engineering - Googl | Roadmap · @BrippneyVargas's Pr | what is fastapi - Search | what is streamlit - Search | FastAPI

https://github.com/users/BrippneyVargas/projects/5/views/4

BrippneyVargas / Projects / @BrippneyVargas's Project Tracking

@BrippneyVargas's Project Tracking — Increased items preview — Feedback

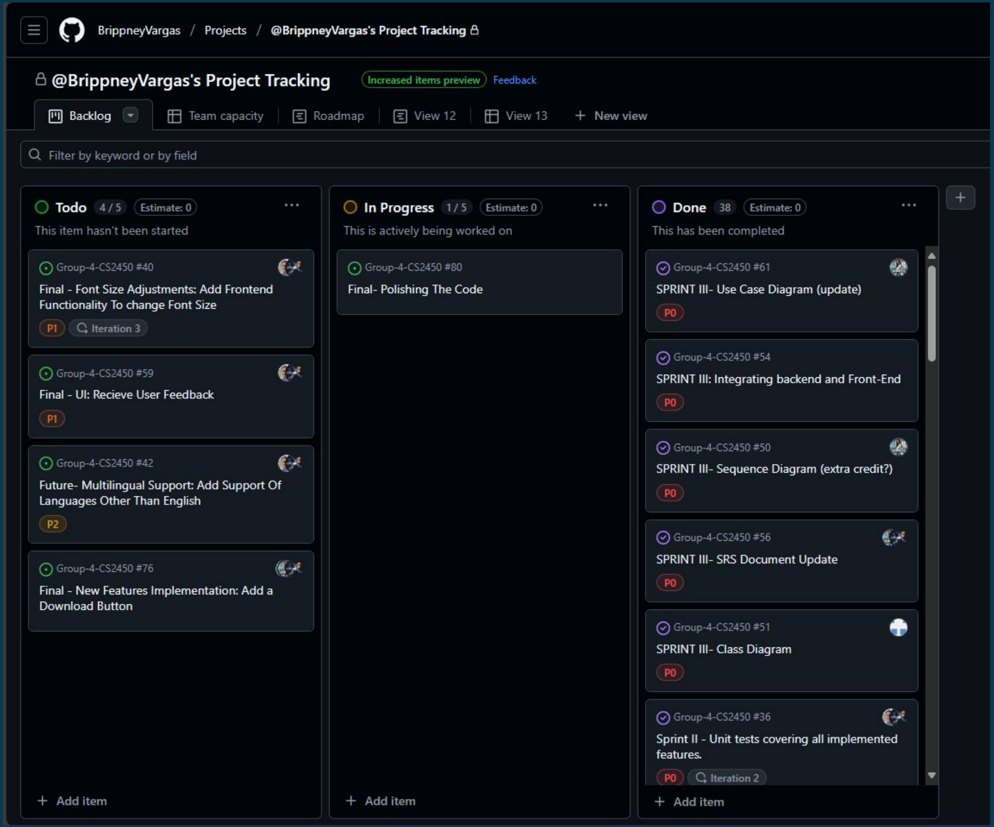
Backlog | Team capacity | Roadmap | View 12 | View 13 | New view

Filter by keyword or by field

March 2025 — April 2025

1. Final- Polishing The Code #80
2. SPRINT V - Documentation: Update SRS #71
3. SPRINT V - Testing: Write Test Cases In Form Of a Table #72
4. SPRINT V - New Features Implementation: Secure Log-in, Authentication #53
5. SPRINT V - New Features Implementation: Implement SQL Database #70
6. SPRINT III- Use Case Diagram (update) #61
7. SPRINT V - New Features Testing: Create Test Report #73
8. Sprint IV - Dark Mode / Light Mode Toggle #39
9. SPRINT III: Integrating backend and Front-End #54
10. SPRINT IV- UI: Pick a name, design logo #63
11. SPRINT IV: UI: Multi-page Web #58
12. Sprint II - Task Organization (Tagging & Categorization). Story Point: 5 #33
13. SPRINT III- UI: add multiple task and view them in tabular mode #52
14. SPRINT III- Sequence Diagram (extra credit?) #50
15. SPRINT V - Meeting Notes: Upload Meeting Notes for 4/2/25 #69
16. SPRINT III- SRS Document Update #56

You can use Control + Space to add an item



BrippneyVargas / Projects / @BrippneyVargas's Project Tracking

@BrippneyVargas's Project Tracking — Increased items preview — Feedback

Backlog | Team capacity | Roadmap | View 12 | View 13 | New view

Filter by keyword or by field

Todo 4 / 5 — Estimate: 0
This item hasn't been started

- Group-4-CS2450 #40 — Final - Font Size Adjustments: Add Frontend Functionality To change Font Size — P1 — Iteration 3
- Group-4-CS2450 #59 — Final - UI: Recieve User Feedback — P1
- Group-4-CS2450 #42 — Future- Multilingual Support: Add Support Of Languages Other Than English — P2
- Group-4-CS2450 #76 — Final - New Features Implementation: Add a Download Button

Add item

In Progress 1 / 5 — Estimate: 0
This is actively being worked on

- Group-4-CS2450 #80 — Final- Polishing The Code

Add item

Done 38 — Estimate: 0
This has been completed

- Group-4-CS2450 #61 — SPRINT III- Use Case Diagram (update) — P0
- Group-4-CS2450 #54 — SPRINT III: Integrating backend and Front-End — P0
- Group-4-CS2450 #50 — SPRINT III- Sequence Diagram (extra credit?) — P0
- Group-4-CS2450 #56 — SPRINT III- SRS Document Update — P0
- Group-4-CS2450 #51 — SPRINT III- Class Diagram — P0
- Group-4-CS2450 #36 — Sprint II - Unit tests covering all implemented features. — P0 — Iteration 2

Add item

Analysis

High-Level Analysis

Functional Requirements	Non-Functional Requirements
Add tags to tasks (Exam, Project, Labwork, Other)	Simple, user-friendly, aesthetic UI
Set priority for each task	Support theme switch (light/dark mode)
Update the task priority status	The system maintain consistent design across different machines
Display the task automatically once successfully added	The system responds quickly when adding, deleting and updating the tasks.
Allow users to add multiple tasks	The system loads task within a few seconds.

Low-Level Analysis

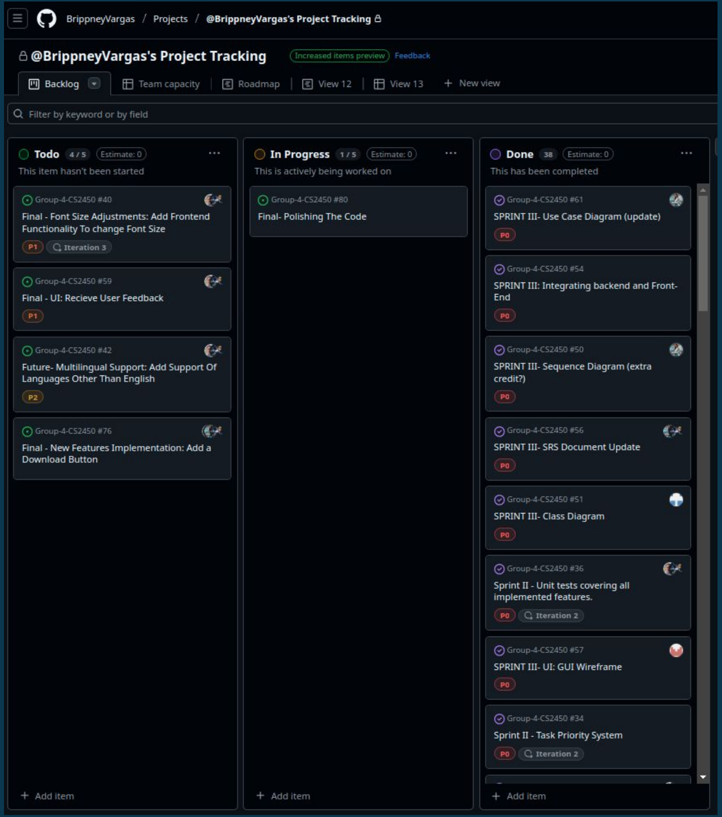
Functional Requirements	Non-Functional Requirements
Set priority for each task	Deleting a task response within 1 second.
View a specific number of tasks (10) per pages	Delete a task with just one click on the delete button
Delete a Task	Load dashboard within 1 seconds.
Duplicate task (same title) is not allowed and will alert the users about it.	Load task from the database within 3 seconds.
Save and load task to the database	Ensure privacy and user authentication.

Development

Development

Language/Framework	Design Principles	Design Pattern
● Python ● Streamlit ● SQLite	● Modularity ● Single Responsibility Principle (SRP) ● Low coupling ● High cohesion ● Interface segregation principle (ISP)	● Model-View-Controller ● Singleton ● Decorator

Project Workflow



BrippneyVargas / Projects / @BrippneyVargas's Project Tracking

@BrippneyVargas's Project Tracking Increased items preview Feedback

Backlog | Team capacity | Roadmap | View 12 | View 13 | + New view

Filter by keyword or by field

Todo 4 / 5 Estimate: 0
This item hasn't been started

- Group-4-CS2450 #40
  Final - Font Size Adjustments: Add Frontend Functionality To change Font Size
  P1 Iteration 3
- Group-4-CS2450 #59
  Final - UI: Recieve User Feedback
  P1
- Group-4-CS2450 #42
  Future- Multilingual Support: Add Support Of Languages Other Than English
  P2
- Group-4-CS2450 #76
  Final - New Features Implementation: Add a Download Button

+ Add item

In Progress 1 / 5 Estimate: 0
This is actively being worked on

- Group-4-CS2450 #80
  Final- Polishing The Code

+ Add item

Done 38 Estimate: 0
This has been completed

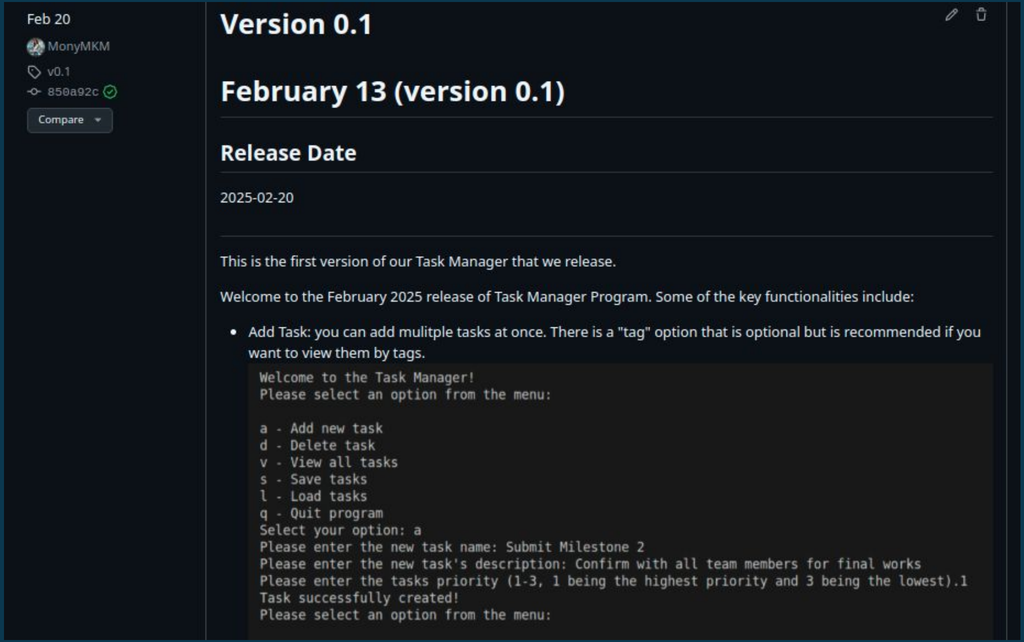
- Group-4-CS2450 #61
  SPRINT III- Use Case Diagram (update)
  P0
- Group-4-CS2450 #54
  SPRINT III: Integrating backend and Front-End
  P0
- Group-4-CS2450 #50
  SPRINT III- Sequence Diagram (extra credit?)
  P0
- Group-4-CS2450 #56
  SPRINT III- SRS Document Update
  P0
- Group-4-CS2450 #51
  SPRINT III- Class Diagram
  P0
- Group-4-CS2450 #36
  Sprint II - Unit tests covering all implemented features.
  P0 Iteration 2
- Group-4-CS2450 #57
  SPRINT III- UI: GUI Wireframe
  P0
- Group-4-CS2450 #34
  Sprint II - Task Priority System
  P0 Iteration 2

+ Add item

- Product Backlog: GitHub Project
- Team Communication:
  - Channel: MS Teams
  - Team Meeting: At least once a week

Versions

- There are 4 versions we have released so far.



Feb 20

MonyMKM

v0.1

850a92c

Version 0.1

February 13 (version 0.1)

Release Date

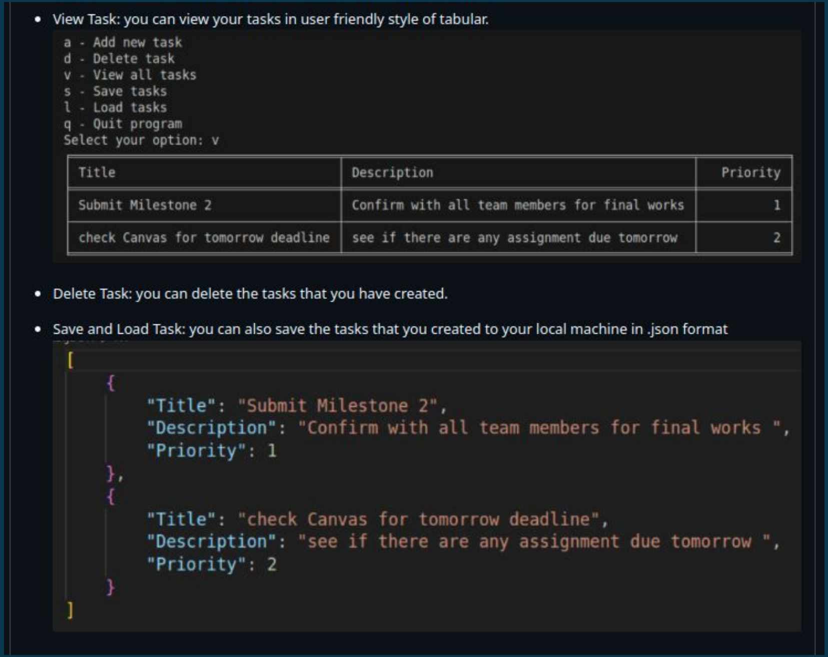
2025-02-20

This is the first version of our Task Manager that we release.

Welcome to the February 2025 release of Task Manager Program. Some of the key functionalities include:

- Add Task: you can add mulitple tasks at once. There is a "tag" option that is optional but is recommended if you want to view them by tags.

```
Welcome to the Task Manager!
Please select an option from the menu:

a - Add new task
d - Delete task
v - View all tasks
s - Save tasks
l - Load tasks
q - Quit program
Select your option: a
Please enter the new task name: Submit Milestone 2
Please enter the new task's description: Confirm with all team members for final works
Please enter the tasks priority (1-3, 1 being the highest priority and 3 being the lowest).1
Task successfully created!
Please select an option from the menu:
```



- View Task: you can view your tasks in user friendly style of tabular.

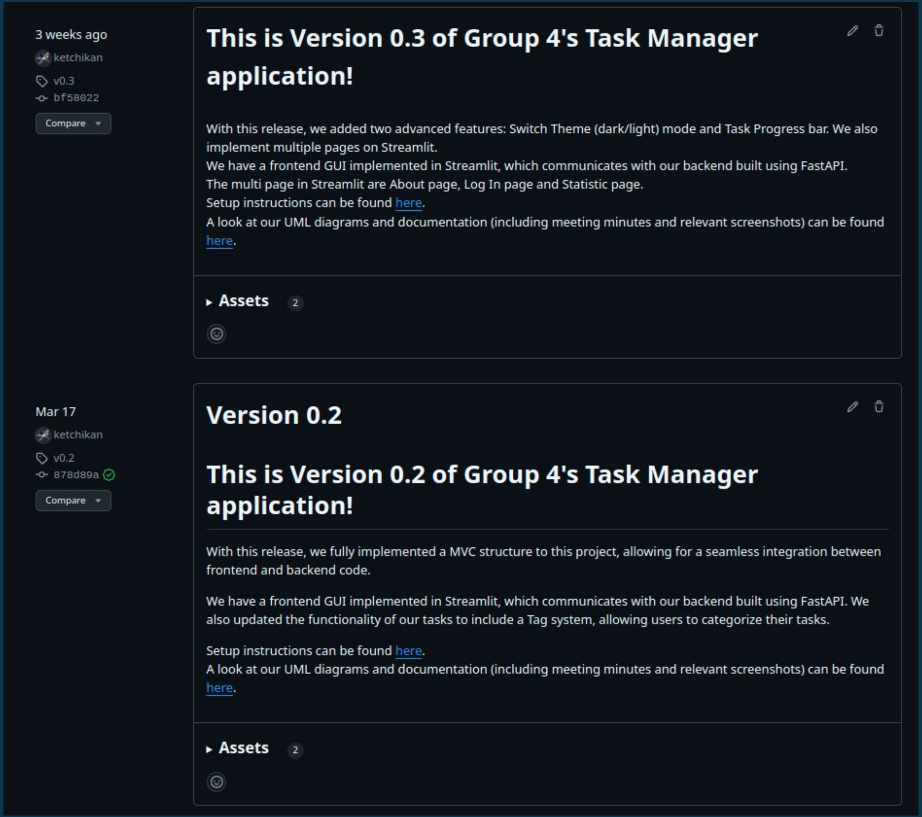
```
a - Add new task
d - Delete task
v - View all tasks
s - Save tasks
l - Load tasks
q - Quit program
Select your option: v
┌─────────────────────────────────┬──────────────────────────────────────────────┬──────────┐
│ Title                           │ Description                                  │ Priority │
├─────────────────────────────────┼──────────────────────────────────────────────┼──────────┤
│ Submit Milestone 2              │ Confirm with all team members for final works│        1 │
├─────────────────────────────────┼──────────────────────────────────────────────┼──────────┤
│ check Canvas for tomorrow deadline │ see if there are any assignment due tomorrow │     2 │
└─────────────────────────────────┴──────────────────────────────────────────────┴──────────┘
```

- Delete Task: you can delete the tasks that you have created.

- Save and Load Task: you can also save the tasks that you created to your local machine in .json format

```
[
    {
        "Title": "Submit Milestone 2",
        "Description": "Confirm with all team members for final works ",
        "Priority": 1
    },
    {
        "Title": "check Canvas for tomorrow deadline",
        "Description": "see if there are any assignment due tomorrow ",
        "Priority": 2
    }
]
```

Versions (Con.)



3 weeks ago
ketchikan
v0.3
bf58022

This is Version 0.3 of Group 4's Task Manager application!

With this release, we added two advanced features: Switch Theme (dark/light) mode and Task Progress bar. We also implement multiple pages on Streamlit.
We have a frontend GUI implemented in Streamlit, which communicates with our backend built using FastAPI.
The multi page in Streamlit are About page, Log In page and Statistic page.
Setup instructions can be found here.
A look at our UML diagrams and documentation (including meeting minutes and relevant screenshots) can be found here.

▸ Assets 2

Mar 17
ketchikan
v0.2
878d89a

Version 0.2

This is Version 0.2 of Group 4's Task Manager application!

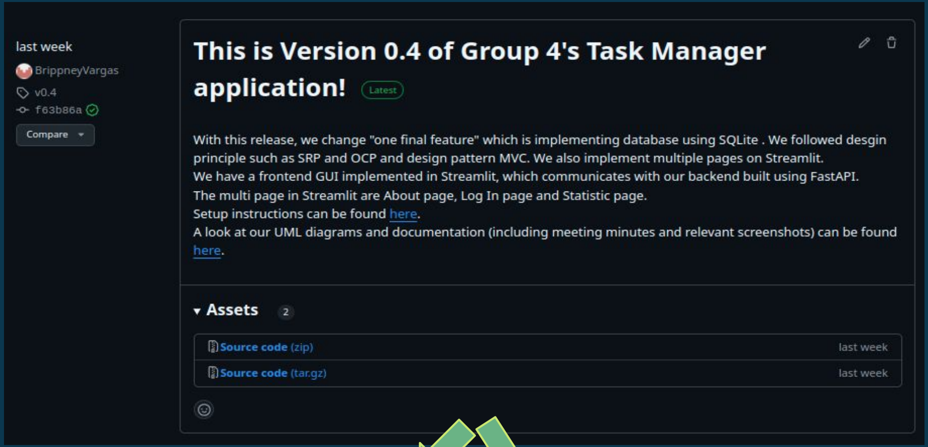
With this release, we fully implemented a MVC structure to this project, allowing for a seamless integration between frontend and backend code.

We have a frontend GUI implemented in Streamlit, which communicates with our backend built using FastAPI. We also updated the functionality of our tasks to include a Tag system, allowing users to categorize their tasks.

Setup instructions can be found here.
A look at our UML diagrams and documentation (including meeting minutes and relevant screenshots) can be found here.

▸ Assets 2

Current Version



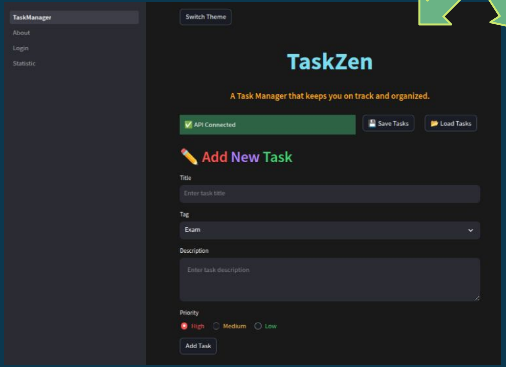
last week
BrippneyVargas
v0.4
f63b86a

This is Version 0.4 of Group 4's Task Manager application! Latest

With this release, we change "one final feature" which is implementing database using SQLite . We followed desgin principle such as SRP and OCP and design pattern MVC. We also implement multiple pages on Streamlit.
We have a frontend GUI implemented in Streamlit, which communicates with our backend built using FastAPI.
The multi page in Streamlit are About page, Log In page and Statistic page.
Setup instructions can be found here.
A look at our UML diagrams and documentation (including meeting minutes and relevant screenshots) can be found here.

▾ Assets 2

Source code (zip)	last week
Source code (tar.gz)	last week



TaskZen

A Task Manager that keeps you on track and organized.

✅ API Connected

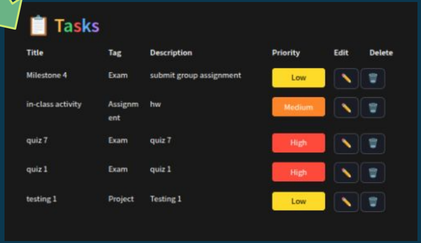
✏️ Add New Task

Title: Enter task title

Tag: Exam

Description: Enter task description

Priority: ● High ○ Medium ○ Low

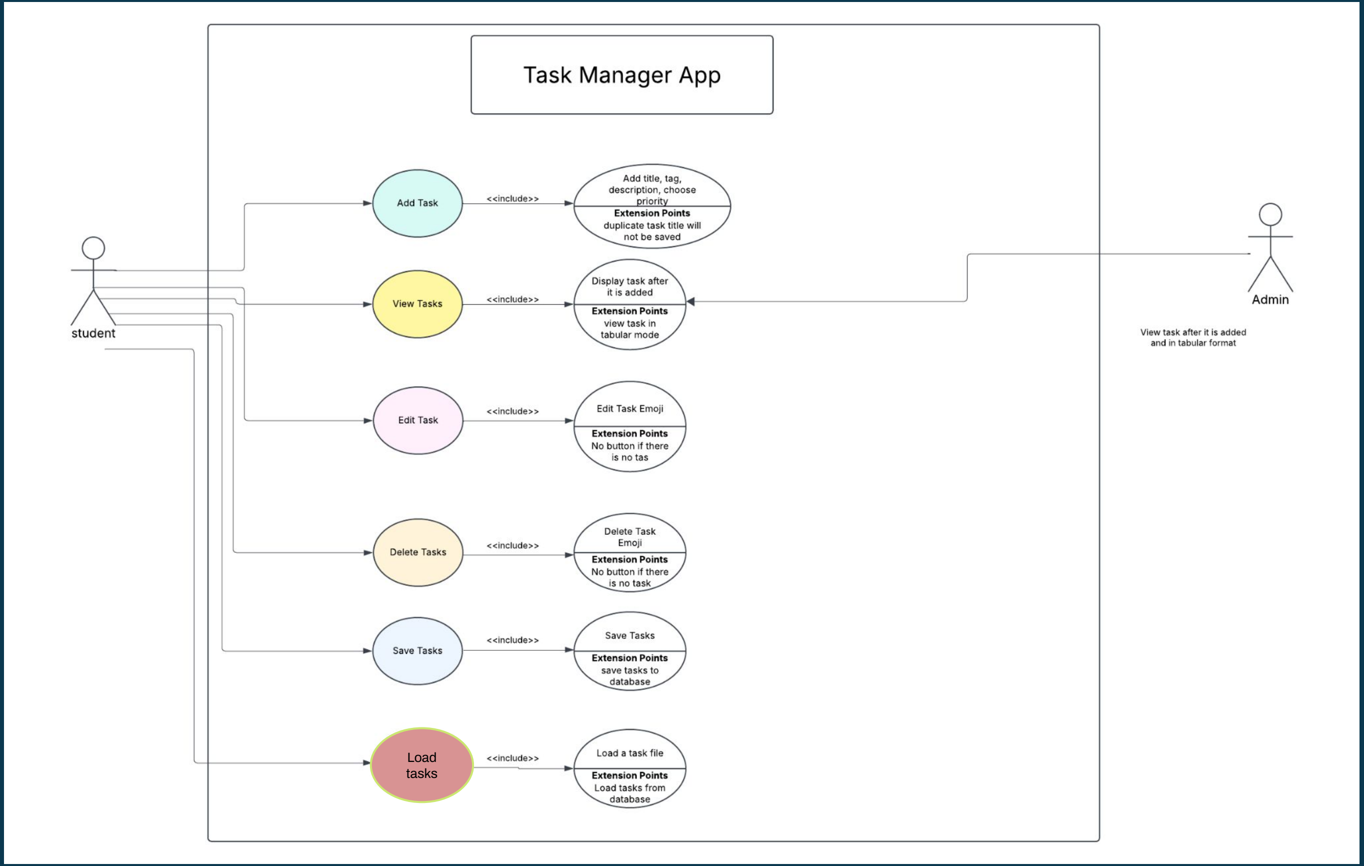


📋 Tasks

Title	Tag	Description	Priority	Edit	Delete
Milestone 4	Exam	submit group assignment	Low	✏️	🗑
in-class activity	Assignment	hw	Medium	✏️	🗑
quiz 7	Exam	quiz 7	High	✏️	🗑
quiz 1	Exam	quiz 1	High	✏️	🗑
testing 1	Project	Testing 1	Low	✏️	🗑

UML DIAGRAMS

Use Case Diagram



Task Manager App

student

Admin

Add Task

<<include>>

Add title, tag, description, choose priority

Extension Points

duplicate task title will not be saved

View Tasks

<<include>>

Display task after it is added

Extension Points

view task in tabular mode

Edit Task

<<include>>

Edit Task Emoji

Extension Points

No button if there is no tas

Delete Tasks

<<include>>

Delete Task Emoji

Extension Points

No button if there is no task

Save Tasks

<<include>>

Save Tasks

Extension Points

save tasks to database

Load tasks

<<include>>

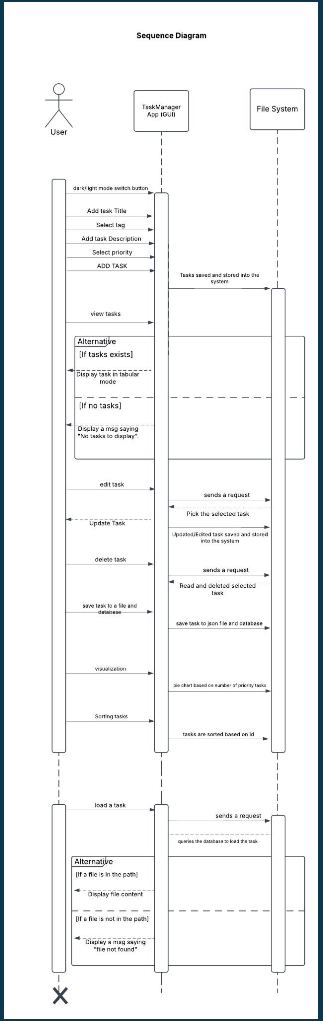
Load a task file

Extension Points

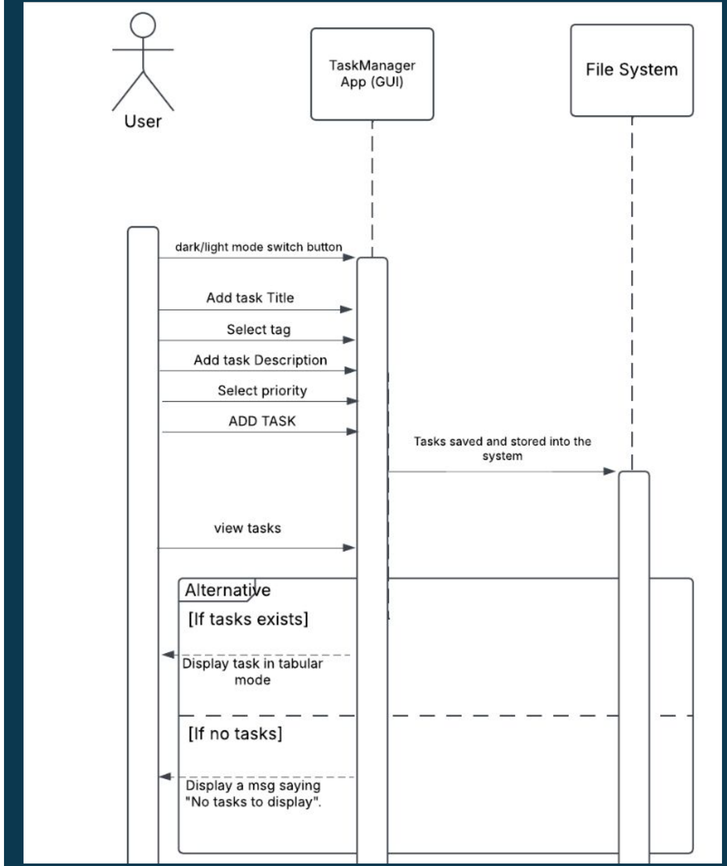
Load tasks from database

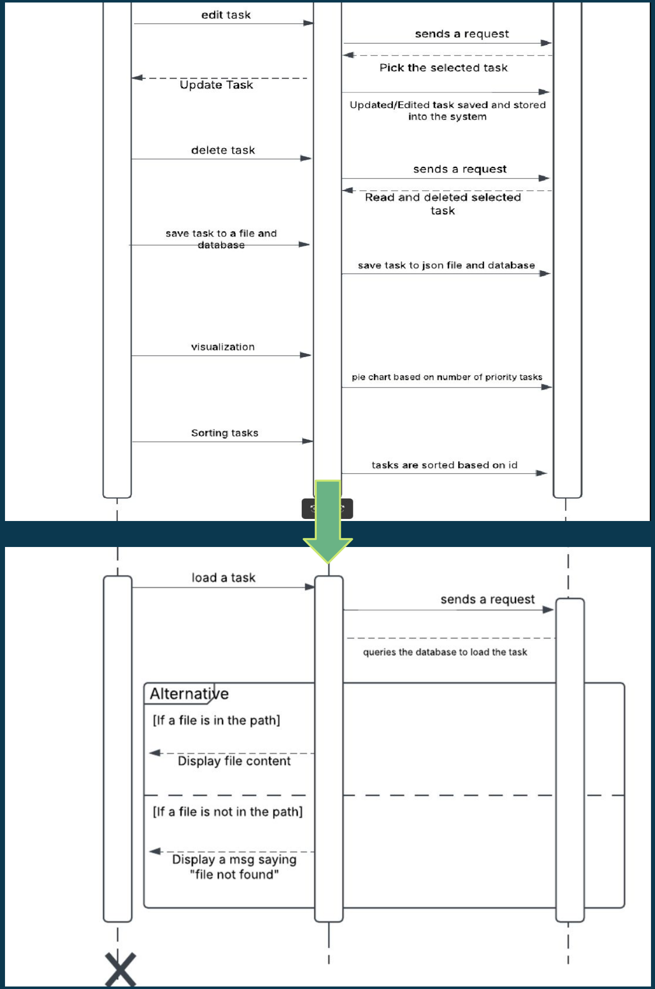
View task after it is added and in tabular format

Sequence Diagram



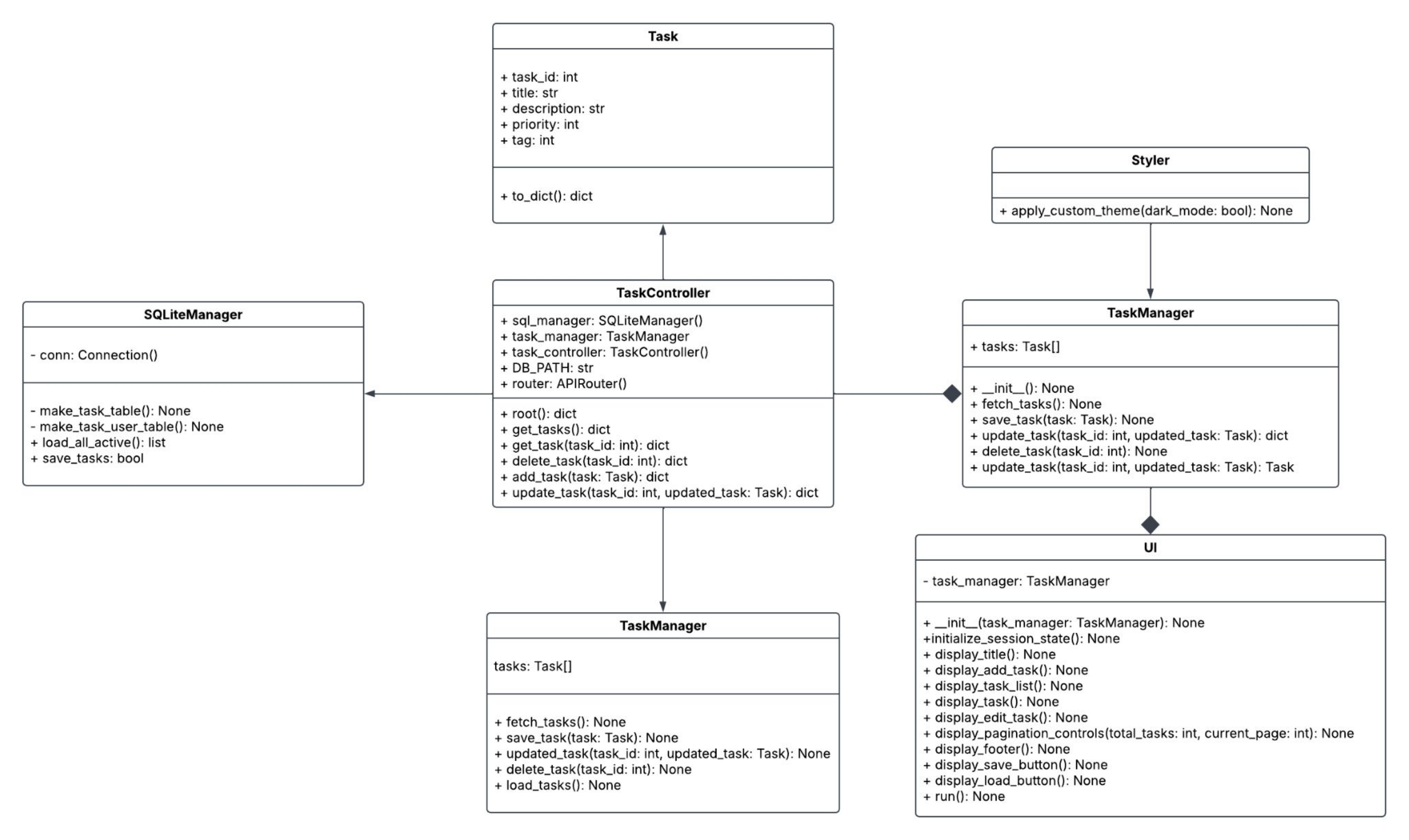
Sequence Diagram
User
TaskManager App (GUI)
File System
dark/light mode switch button
Add task Title
Select tag
Add task Description
Select priority
ADD TASK
Tasks saved and stored into the system
view tasks
Alternative
[If tasks exists]
Display task in tabular mode
[If no tasks]
Display a msg saying "No tasks to display".
edit task
sends a request
Pick the selected task
Update Task
Updated/Edited task saved and stored into the system
delete task
sends a request
Read and deleted selected task
save task to a file and database
save task to json file and database
visualization
pie chart based on number of priority tasks
Sorting tasks
tasks are sorted based on id
load a task
sends a request
queries the database to load the task
Alternative
[If a file is in the path]
Display file content
[If a file is not in the path]
Display a msg saying "file not found"





edit task
sends a request
Pick the selected task
Update Task
Updated/Edited task saved and stored into the system
delete task
sends a request
Read and deleted selected task
save task to a file and database
save task to json file and database
visualization
pie chart based on number of priority tasks
Sorting tasks
tasks are sorted based on id
load a task
sends a request
queries the database to load the task
Alternative
[If a file is in the path]
Display file content
[If a file is not in the path]
Display a msg saying "file not found"

Class Diagram



Task

+ task_id: int
+ title: str
+ description: str
+ priority: int
+ tag: int

+ to_dict(): dict

Styler

+ apply_custom_theme(dark_mode: bool): None

SQLiteManager

- conn: Connection()

- make_task_table(): None
- make_task_user_table(): None
+ load_all_active(): list
+ save_tasks: bool

TaskController

+ sql_manager: SQLiteManager()
+ task_manager: TaskManager
+ task_controller: TaskController()
+ DB_PATH: str
+ router: APIRouter()

+ root(): dict
+ get_tasks(): dict
+ get_task(task_id: int): dict
+ delete_task(task_id: int): dict
+ add_task(task: Task): dict
+ update_task(task_id: int, updated_task: Task): dict

TaskManager

+ tasks: Task[]

+ __init__(): None
+ fetch_tasks(): None
+ save_task(task: Task): None
+ update_task(task_id: int, updated_task: Task): dict
+ delete_task(task_id: int): None
+ update_task(task_id: int, updated_task: Task): Task

UI

- task_manager: TaskManager

+ __init__(task_manager: TaskManager): None
+initialize_session_state(): None
+ display_title(): None
+ display_add_task(): None
+ display_task_list(): None
+ display_task(): None
+ display_edit_task(): None
+ display_pagination_controls(total_tasks: int, current_page: int): None
+ display_footer(): None
+ display_save_button(): None
+ display_load_button(): None
+ run(): None

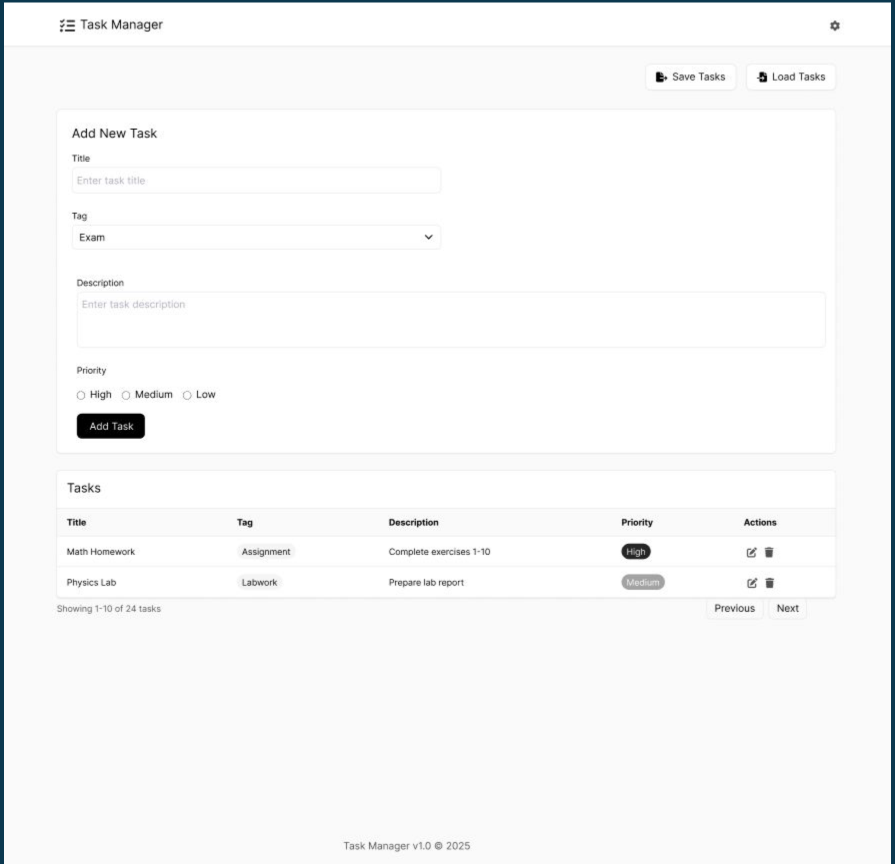
TaskManager

tasks: Task[]

+ fetch_tasks(): None
+ save_task(task: Task): None
+ updated_task(task_id: int, updated_task: Task): None
+ delete_task(task_id: int): None
+ load_tasks(): None

Wireframes

Initial Wireframe



Add New Task

Title

Enter task title

Tag

Exam

Description

Enter task description

Priority

○ High ○ Medium ○ Low

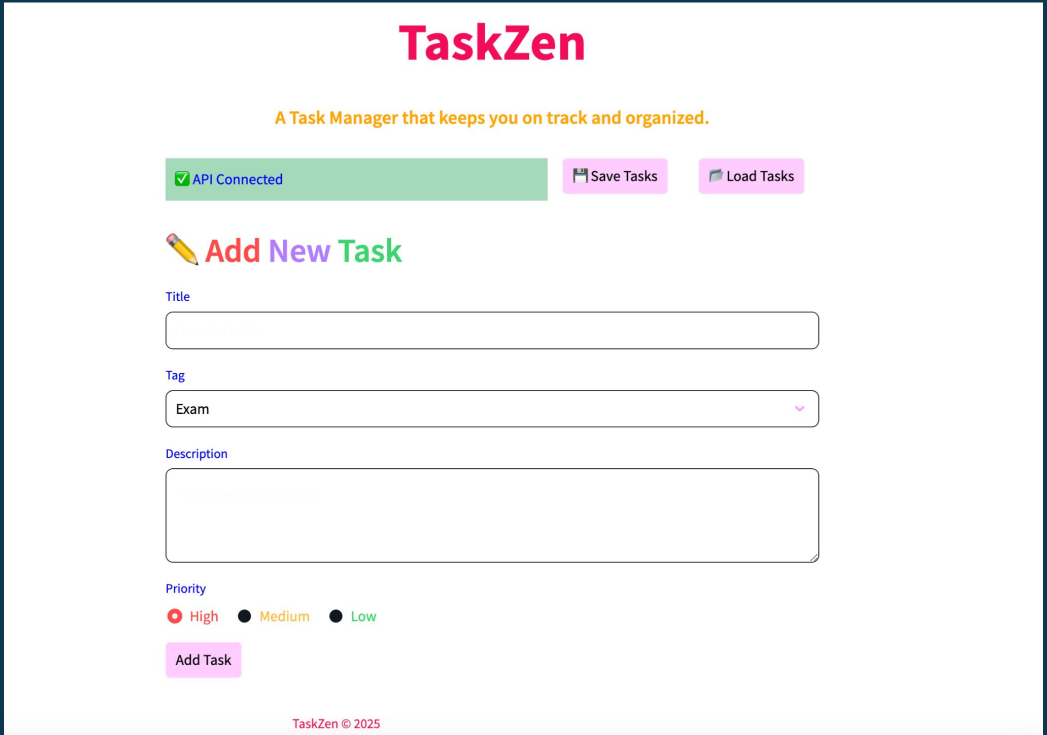
Tasks

Title	Tag	Description	Priority	Actions
Math Homework	Assignment	Complete exercises 1-10	High	
Physics Lab	Labwork	Prepare lab report	Medium	

Showing 1-10 of 24 tasks

Task Manager v1.0 © 2025

Final Design



TaskZen

A Task Manager that keeps you on track and organized.

✅ API Connected

💾 Save Tasks

📁 Load Tasks

✏️ Add New Task

Title

Enter task title

Tag

Exam

Description

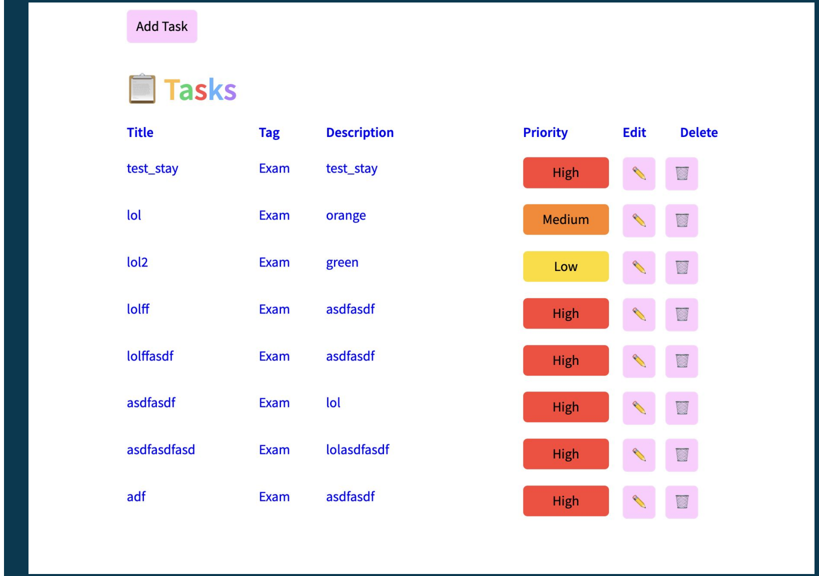
Enter task description

Priority

● High ● Medium ● Low

Add Task

TaskZen © 2025



Add Task

📋 Tasks

Title	Tag	Description	Priority	Edit	Delete
test_stay	Exam	test_stay	High	✏️	🗑
lol	Exam	orange	Medium	✏️	🗑
lol2	Exam	green	Low	✏️	🗑
lolff	Exam	asdfasdf	High	✏️	🗑
lolffasdf	Exam	asdfasdf	High	✏️	🗑
asdfasdf	Exam	lol	High	✏️	🗑
asdfasdfasd	Exam	lolasdfasdf	High	✏️	🗑
adf	Exam	asdfasdf	High	✏️	🗑

Features

Current Features

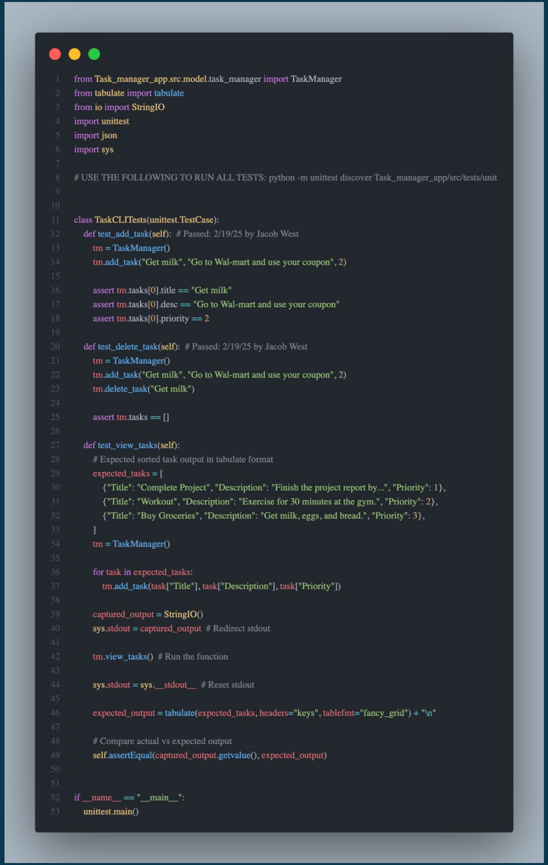
- Creating, editing, and completing tasks
- User Authentication - keep your tasks to yourself!
- SQL Database
- Dark Mode
- Statistics

Future Features

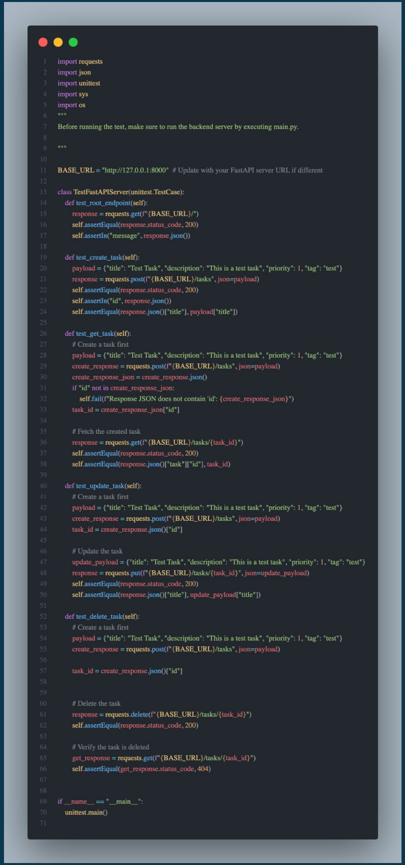
- Implement AI-driven task suggestions based on user behavior and priorities.
- Use natural language processing to allow users to create tasks via voice or chat.
- Introduce automated reminders and follow-ups based on task deadlines and urgency.
- Offer end-to-end encryption for sensitive task data.
- Provide automatic daily cloud backups.

Testing

Unittesting

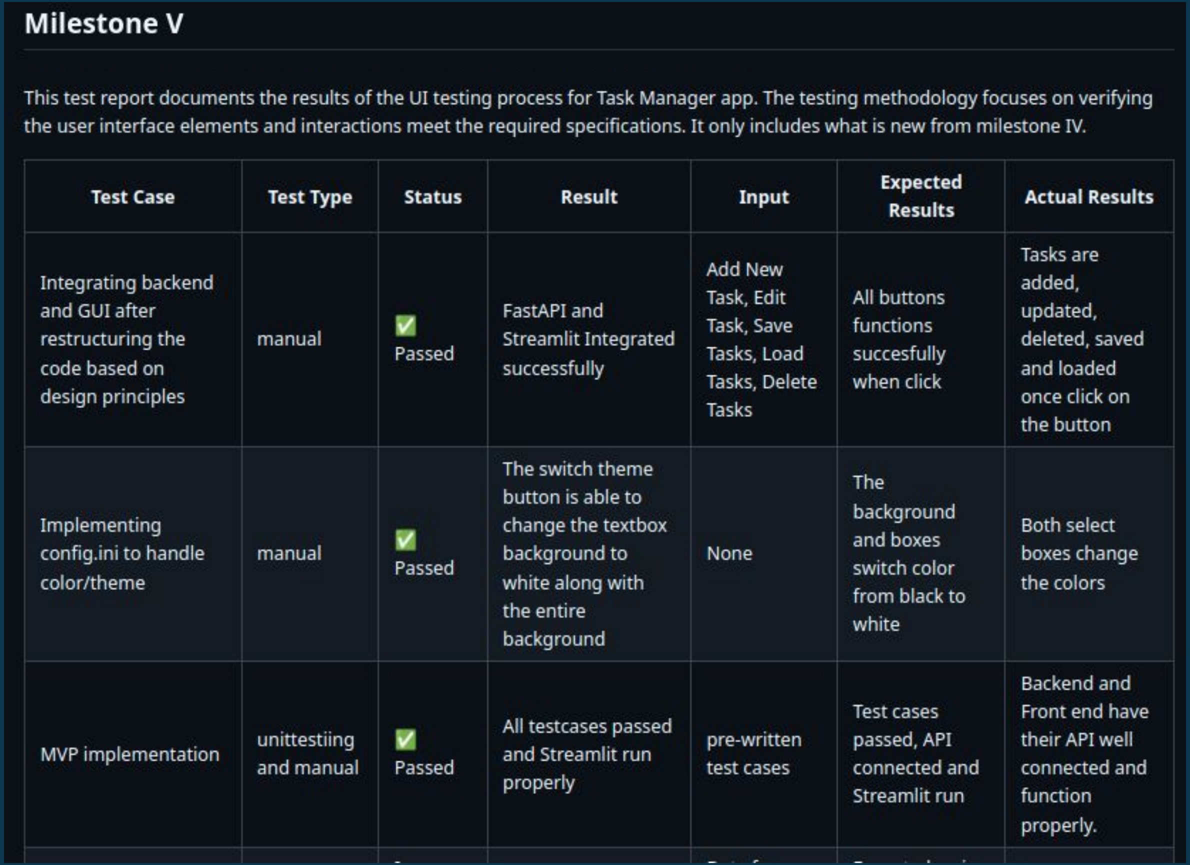


```python
from Task_manager_app.src.model.task_manager import TaskManager
from tabulate import tabulate
from io import StringIO
import unittest
import json
import sys

# USE THE FOLLOWING TO RUN ALL TESTS: python -m unittest discover Task_manager_app/src/tests/unit


class TaskCLITests(unittest.TestCase):
    def test_add_task(self):  # Passed: 2/19/25 by Jacob West
        tm = TaskManager()
        tm.add_task("Get milk", "Go to Wal-mart and use your coupon", 2)

        assert tm.tasks[0].title == "Get milk"
        assert tm.tasks[0].desc == "Go to Wal-mart and use your coupon"
        assert tm.tasks[0].priority == 2

    def test_delete_task(self):  # Passed: 2/19/25 by Jacob West
        tm = TaskManager()
        tm.add_task("Get milk", "Go to Wal-mart and use your coupon", 2)
        tm.delete_task("Get milk")

        assert tm.tasks == []

    def test_view_tasks(self):
        # Expected sorted task output in tabulate format
        expected_tasks = [
            {"Title": "Complete Project", "Description": "Finish the project report by...", "Priority": 1},
            {"Title": "Workout", "Description": "Exercise for 30 minutes at the gym.", "Priority": 2},
            {"Title": "Buy Groceries", "Description": "Get milk, eggs, and bread.", "Priority": 3},
        ]
        tm = TaskManager()

        for task in expected_tasks:
            tm.add_task(task["Title"], task["Description"], task["Priority"])

        captured_output = StringIO()
        sys.stdout = captured_output  # Redirect stdout

        tm.view_tasks()  # Run the function

        sys.stdout = sys.__stdout__  # Reset stdout

        expected_output = tabulate(expected_tasks, headers="keys", tablefmt="fancy_grid") + "\n"

        # Compare actual vs expected output
        self.assertEqual(captured_output.getvalue(), expected_output)


if __name__ == "__main__":
    unittest.main()
```



```python
import requests
import json
import unittest
import sys
import os
"""
Before running the test, make sure to run the backend server by executing main.py.

"""

BASE_URL = "http://127.0.0.1:8000"  # Update with your FastAPI server URL if different

class TestFastAPIServer(unittest.TestCase):
    def test_root_endpoint(self):
        response = requests.get(f"{BASE_URL}/")
        self.assertEqual(response.status_code, 200)
        self.assertIn("message", response.json())

    def test_create_task(self):
        payload = {"title": "Test Task", "description": "This is a test task", "priority": 1, "tag": "test"}
        response = requests.post(f"{BASE_URL}/tasks", json=payload)
        self.assertEqual(response.status_code, 200)
        self.assertIn("id", response.json())
        self.assertEqual(response.json()["title"], payload["title"])

    def test_get_task(self):
        # Create a task first
        payload = {"title": "Test Task", "description": "This is a test task", "priority": 1, "tag": "test"}
        create_response = requests.post(f"{BASE_URL}/tasks", json=payload)
        create_response_json = create_response.json()
        if "id" not in create_response_json:
            self.fail(f"Response JSON does not contain 'id': {create_response_json}")
        task_id = create_response_json["id"]

        # Fetch the created task
        response = requests.get(f"{BASE_URL}/tasks/{task_id}")
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.json()["task"]["id"], task_id)

    def test_update_task(self):
        # Create a task first
        payload = {"title": "Test Task", "description": "This is a test task", "priority": 1, "tag": "test"}
        create_response = requests.post(f"{BASE_URL}/tasks", json=payload)
        task_id = create_response.json()["id"]

        # Update the task
        update_payload = {"title": "Test Task", "description": "This is a test task", "priority": 1, "tag": "test"}
        response = requests.put(f"{BASE_URL}/tasks/{task_id}", json=update_payload)
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.json()["title"], update_payload["title"])

    def test_delete_task(self):
        # Create a task first
        payload = {"title": "Test Task", "description": "This is a test task", "priority": 1, "tag": "test"}
        create_response = requests.post(f"{BASE_URL}/tasks", json=payload)

        task_id = create_response.json()["id"]


        # Delete the task
        response = requests.delete(f"{BASE_URL}/tasks/{task_id}")
        self.assertEqual(response.status_code, 200)

        # Verify the task is deleted
        get_response = requests.get(f"{BASE_URL}/tasks/{task_id}")
        self.assertEqual(get_response.status_code, 404)


if __name__ == "__main__":
    unittest.main()

```

Manual Testing (UI)

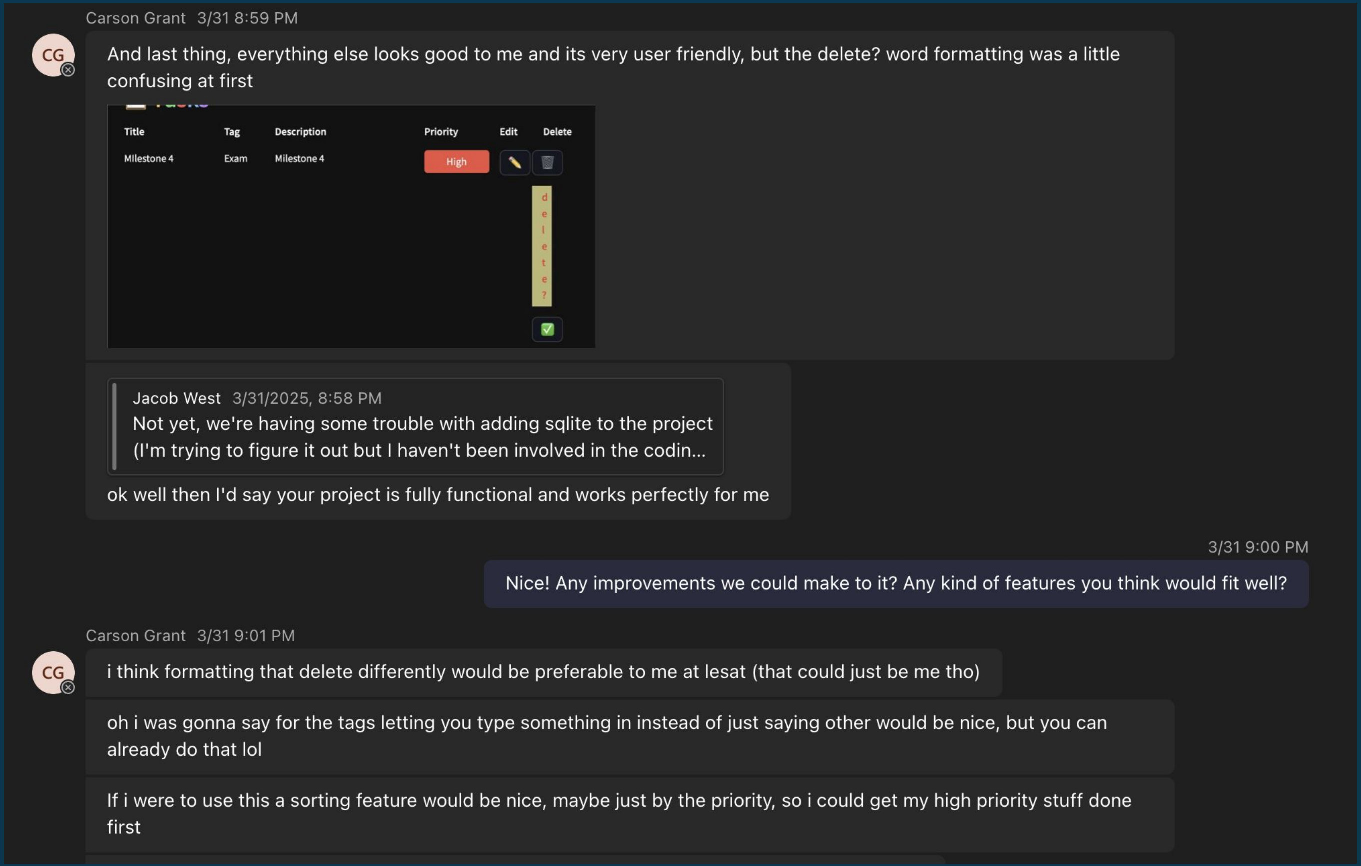


Milestone V

This test report documents the results of the UI testing process for Task Manager app. The testing methodology focuses on verifying the user interface elements and interactions meet the required specifications. It only includes what is new from milestone IV.

Test Case	Test Type	Status	Result	Input	Expected Results	Actual Results
Integrating backend and GUI after restructuring the code based on design principles	manual	✅ Passed	FastAPI and Streamlit Integrated successfully	Add New Task, Edit Task, Save Tasks, Load Tasks, Delete Tasks	All buttons functions succesfully when click	Tasks are added, updated, deleted, saved and loaded once click on the button
Implementing config.ini to handle color/theme	manual	✅ Passed	The switch theme button is able to change the textbox background to white along with the entire background	None	The background and boxes switch color from black to white	Both select boxes change the colors
MVP implementation	unittestiing and manual	✅ Passed	All testcases passed and Streamlit run properly	pre-written test cases	Test cases passed, API connected and Streamlit run	Backend and Front end have their API well connected and function properly.

Peer Review



Carson Grant 3/31 8:59 PM

And last thing, everything else looks good to me and its very user friendly, but the delete? word formatting was a little confusing at first

Title	Tag	Description	Priority	Edit	Delete
Milestone 4	Exam	Milestone 4	High	✏️	🗑

> Jacob West 3/31/2025, 8:58 PM
> Not yet, we're having some trouble with adding sqlite to the project (I'm trying to figure it out but I haven't been involved in the codin...

ok well then I'd say your project is fully functional and works perfectly for me

3/31 9:00 PM

Nice! Any improvements we could make to it? Any kind of features you think would fit well?

Carson Grant 3/31 9:01 PM

i think formatting that delete differently would be preferable to me at lesat (that could just be me tho)

oh i was gonna say for the tags letting you type something in instead of just saying other would be nice, but you can already do that lol

If i were to use this a sorting feature would be nice, maybe just by the priority, so i could get my high priority stuff done first

DEMO

Thank you

Q & A