

Trello Task Management Documentation

(Backend API's)

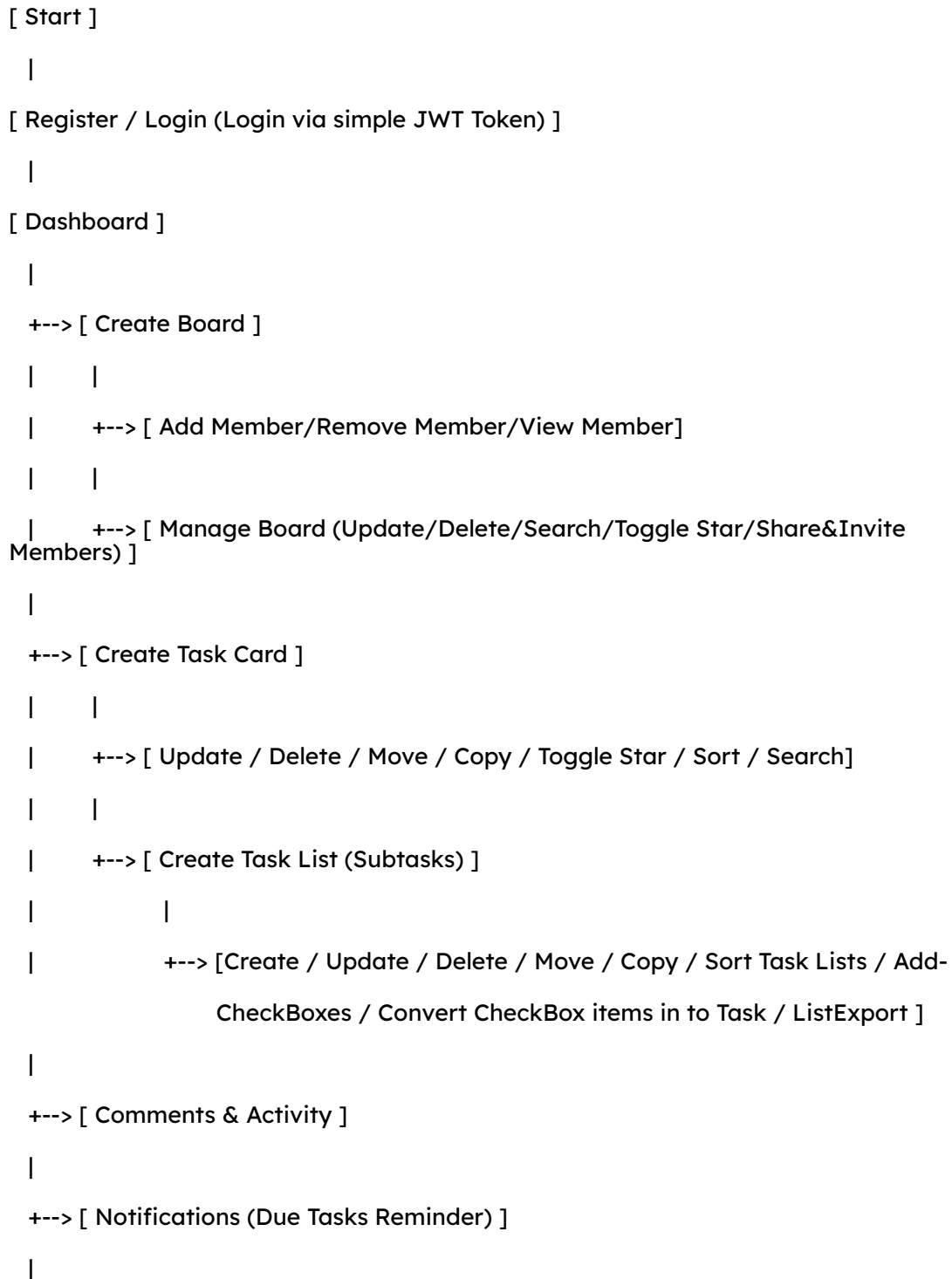
- This is a Function-based API Task.
- This Task management will handle all users' tasks on one Main Task Board and will be able to add/assign their team members to Board and create a Project task environment
- The Trello Task Management allows users to manage boards, tasks, and collaborate with team members.
- The API uses simple JWT Authentication and supports role-based activities(Admin, Member, Super User).
- Backend Tools and Framework used in this project Python(3.12.3), Django(4.2.23), Rest Framework(3.16.6) and MySQL workbench database, Postman (for testing APIs)

Project Folder Structure of Backend:

```
Trello-Task-Management/  
| — manage.py          # Django project manager  
| — requirements.txt    # Dependencies  
| — README.md          # Project overview  
| — LICENSE            # License file  
| — exported log data.json # Exported activity logs  
|  
| — trello_project/     # Main project settings  
|   | — __init__.py  
|   | — settings.py  
|   | — urls.py  
|   | — wsgi.py
```

```
|   └─ asgi.py
|
|
|─ trello_app/      # Core application
|   └─ __init__.py
|   └─ admin.py      # Django admin panel
|   └─ apps.py       # App configuration
|   └─ models.py     # Database models
|   └─ serializers.py # DRF serializers
|   └─ urls.py       # API routes
|
|   └─ views/        # API views
|       └─ board.py  # Board APIs
|       └─ task.py   # Task APIs
|       └─ authentication.py # User & Authentication APIs
|       └─ utils.py  # Collections of helper functions
|       └─ └─ __init__.py
|
|   └─ migrations/   # Django migration files
|
|─ media/            # Media storage
|   └─ exports/      # Exported files
|   └─ task_attachment/ # Task attachments
|   └─ task_images/  # Task images
|
|─ myvenv/           # Virtual environment
|
```

Flow Diagram of Trello Task Management:



[End]

Functions List

→ Total 39 functions divided into Authentication, Board management, Task Management.

★ **Authentication Functions (09)**

1. register_user() - Register a new user
2. login_user() - Login user
3. forgot_password_sent_email() - Send forgot password email
4. reset_password() - Reset user password
5. update_profile() - Update user profile
6. delete_profile() - Delete user profile
7. search_view_all_users() - Search or view all users
8. view_my_profile() - View my profile
9. show_activity() - Show user activity

★ **Board Functions (11)**

10. create_board() - Create a board
11. update_board() - Update a board
12. delete_board() - Delete a board
13. add_member_to_board() - Add member to board
14. remove_member_from_board() - Remove member from board
15. view_board_members() - View board members
16. get_my_board() - Get my boards
17. search_boards() - Search boards

- 18. star_board() - Star a board
- 19. share_invite() - Share invite link
- 20. notifications() - Show notifications

★ TaskCard Functions (07)

- 21. create_task() - Create a task card
- 22. update_task() - Update a task card
- 23. delete_task() - Delete a task card
- 24. search_tasks_by() - Search tasks
- 25. star_task_card() - Star a task card
- 26. copy_task_card() - Copy a task card
- 27. move_task_card_to_other_board() - Move task card to another board

★ TaskList Functions (12)

- 28. create_task_lists() - Create a task list
- 29. update_tasks_lists() - Update a task list
- 30. tasks_lists_delete() - Delete a task list
- 31. copy_task_list() - Copy a task list
- 32. move_task_list() - Move a task list to another board
- 33. sort_task_lists() - Sort task lists
- 37. tasklist_checklist_progress() - Add CheckBoxes in Tasklists and show progress(%) of it.
- 35. convert_checkbox_to_tasklist() - Convert Checkbox's item in to your Task list
- 36. create_comment() - Create a comment
- 37. edit_comment() - Edit a comment

38. delete_comment() – Delete a comment

39. print_export_share() – Print / Export / Share board

API Endpoint Reference

❖ Authentication & User Management

1. register_user()

- Endpoint: POST /register_user/
- Description: Register a new user (username, email, password are required)
- Authentication: Not Required
- Request body: {

```
"username": "vills",    // string (unique username)
"email": "vills@example.com",    // string (valid email)
"password": "StrongPass@123",    // string ( min 8 chars)
"full_name": "Vills Patel",    // string (optional full name)
"profile_image": "profile.jpg"    // string (image file name / URL)
}
```

2. login_user()

- Endpoint: POST /login_user/
- Description: Login user and generate JWT token
- Authentication: Not Required
- Request body:

```
{
"username": "vills",    // string (username used in register)
"password": "StrongPass@123" // string (username used in register)
}
```

3. forgot_password_sent_email()

- Endpoint: POST /forgot_password_sent_email/
- Description: Initiate password reset process and send email
- Authentication: Not Required
- Request body:


```
{
  "email": "vills@example.com" // string (registered email required)
}
```

4. reset_password()

- Endpoint: POST /reset_password/
- Description: Complete password reset with token
- Authentication: Not Required
- Request body:

```
{
  "email": "vills@example.com", // string (registered email)
  "password": "NewPass@123", // string (new strong password)
  "OTP": "458921" // string or int (OTP sent via email)
}
```

5. update_profile()

- Endpoint: POST /update_profile/
- Description: Update current user's profile information
- Authentication:  Required
- Request body:

```
{
  "username": "newvills", // string (updated username)
  "email": "newvills@example.com", // string (updated email)
  "password": "Updated@123", // string (new password)
  "full_name": "Vills Updated", // string (new full name)
  "profile_image": "new_profile.jpg" // string (updated profile pic)
}
```

6. delete_profile()

- Endpoint: POST /delete_profile/
- Description: Delete current user's account
- Authentication: ☒ Required
- Request body:

```
{
  "user_id": 12 // int (logged-in user's ID)
}
```

7. search_view_all_users() - Only Super User can perform this action

- Endpoint: POST /search_view_all_users/
- Description: Search specific users by name/email or view all users
- Authentication: ☒ Required
- Request body:

```
{
  "username": "john", // string (optional filter by username)
  "email": "john@example.com", // string (optional filter by email)
  "full_name": "John Doe" // string (optional filter by full name)
}
```

8. view_my_profile()

- Endpoint: POST /view_my_profile/
- Description: View current user's profile details
- Authentication: ☒ Required
- Request body:

// No request body needed (JWT user context used)

9. show_activity()

- Endpoint: POST /show_activity/
- Description: View recent user activity and actions, Board admins can view his members activity, members or normal user can see his own activity only.
- Authentication: ☒ Required
- Request body:

// No extra body; activity fetched based on logged-in user

❖ Board Management Endpoints

Function	Normal User	Board Member	Board Admin	Super User
Create Board	✓	✓	✓	✓
Delete Board	✗	✗	✓	✓
Search All Users	✗	✗	✗	✓
Add Member to Board	✗	✗	✓	✓

10. create_board() – Create a board

- Endpoint: POST /create_board/
- Description: Create a new board
- Authentication: ✓ Required
- Request body:

```
{  
  "title": "Project Alpha", // string (board name)  
  "description": "Board for managing Alpha tasks", // string  
  "visibility": "private", // string ("private" or "public")  
  "members": ['user@gmail.com', 'user2@gmail.com', 'user3@gmail.com'] // list of  
  emails in string  
}
```

11. update_board() Only Board Admin can perform this action

- Endpoint: POST /update_board/
- Description: Update board details

- Authentication: ☒ Required
- Request body:

```
{
  "board_id": 1, // int
  "title": "Project Alpha Updated", // string
  "description": "Updated board description", // string
  "visibility": "public" // string
}
```

12. delete_board() Only Board Admin can perform this action

- Endpoint: POST /delete_board/
- Description: Delete a specific board
- Authentication: ☒ Required
- Request body:

```
{
  "board_id": 1 // int
}
```

13. add_member_to_board() Only Board Admin can perform this action

- Endpoint: POST /add_member_to_board/
- Description: Add user to board as a member via enter board id and email of user's
- Authentication: ☒ Required
- Request body:

```
{
  "board_id": 1, // int(board ID)
  "email": "newuser@example.com" // string (user email)
}
```

14. remove_member_from_board() Only Board Admin can perform this action

- Endpoint: POST /remove_member_from_board/

- Description: Remove user to board as a member via enter board id and email of user's
- Authentication: ☒ Required
- Request body:

```
{
    "board_id": 1,
    "email": "removeuser@example.com"
}
```

15. view_board_members()

- Endpoint: POST /view_board_members/
- Description: List all members of a specific board
- Authentication: ☒ Required
- Request body:

```
{
    "board_id": 1 // int (board ID)
}
```

16. get_my_board()

Users can perform Get data, Sort, Search, operation in this function.

- Endpoint: POST /get_my_board/
- Description: View board with all tasks and details
- Authentication: ☒ Required
- Request body:


```
{
    "board_id": 1,
    "board_title": "Project Alpha", // string
    "board_description": "Main project board for tracking tasks", // string
    "no_members": 5, // int
    "complated": false, // boolean
    "task_title": "Design Homepage", // string
}
```

```

"task_description": "Create UI for homepage", // string
"created_by": 10, // int (user_id)
"tasklist_title": "UI Tasks", // string
"tasklist_description": "Frontend related tasks", // string
"priority": "High", // string
"label_color": "#FF5733", // string (hex code) or ("red")
"start_date": "2025-08-30", // string
"due_date": "2025-09-05", // string (null if not set)
"is_completed": false, // boolean (0/1 int)
"updated_by": 102, // int (or "102" string)
"assigned_to": [103, 104], // list of int
"is_starred": true, // boolean
"no_due": false, // boolean
"overdue": true, // boolean
"due_today": false, // boolean
"due_tomorrow": true, // boolean
"due_next_week": false, // boolean
"due_on_this_month": true, // boolean
"due_on_this_week": false // boolean
}

```

17. search_boards()

- Endpoint: POST /search_boards/
- Description: Search and filter boards by name or criteria
- Authentication:  Required
- Request body:

```

{
"board_id": 1, // int
"board_title": "Project Alpha", // string

```

```
"board_description": "Alpha Board", // string

"visibility": "public" // string

}
```

18. star_board()

- Endpoint: POST /star_board/
- Description: Mark a board as favorite/starred for quick access
- Authentication: ☒ Required
- Request body:

```
{

"board_id": 1, // int

"is_starred": true // boolean

}
```

19. share_invite()

- Endpoint: POST /share_invite/
- Description: Generate a shareable link or invite for a board
- Authentication: ☒ Required
- Request body:

```
{

"board_id": 1,

"email": "inviteuser@example.com" // string

}
```

20. notifications()

- Endpoint: POST /notifications/
- Description: Retrieve all notifications related to the user's due tasks before 2 days. This setup is temporary for backend and practice.
- Authentication: ☒ Required
- Request body:

// No body needed (fetch user notifications automatically)

❖ Task Management Endpoints

21. create_task()

- Endpoint: POST /create_task/
- Description: Create a new task card under a board
- Authentication: ☒ Required
- Request body:

```
{  
  
  "title": "Design Homepage", // string  
  
  "description": "Create a responsive homepage design", // string  
  
  "board_id": 1 // int  
  
}
```

22. update_task()

- Endpoint: POST /update_task/
- Description: Update task card details
- Authentication: ☒ Required
- Request body:

```
{  
  
  "task_id": 10, // int  
  
  "title": "Design Homepage Updated", // string (optional)  
  
  "description": "Updated description for homepage", (optional)  
  
  "board_id": 1,  
  
  "is_completed": false (optional)  
  
}
```

23. delete_task()


- Endpoint: POST /delete_task/
- Description: Delete a specific task card
- Authentication: ☒ Required
- Request body:

```
{
```

```
"task_id": 10 // int

}
```

24. search_tasks_by()

- Endpoint: POST /search_tasks_by/
- Description: Search tasks by keyword in title/description
- Authentication:  Required
- Request body:


```
{

"title": "Design", // string (keyword)

"is_starred": true // boolean

}
```

25. star_task_card()


- Endpoint: POST /star_task_card/
- Description: Mark a task card as starred/favorite
- Authentication:  Required
- Request body:

```
{

"task_id": 10

}
```

26. copy_task_card()

- Endpoint: POST /copy_task_card/
- Description: Create a duplicate of an existing task card within a board
- Authentication:  Required
- Request body:


```
{

"task_id": 10

}
```

27. move_task_card_to_other_board()


- Endpoint: POST /move_task_card_to_other_board/

- Description: Move a task card from one board to another board
- Authentication:  Required
- Request body:

```
{
  "task_id": 10,
  "new_board_id": 2
}
```

❖ Task List Management Endpoints

28. create_task_lists()

- Endpoint: POST /create_task_lists/
- Description: Create a new task list (subtask) under a task card
- Authentication:  Required
- Request body:

```
{
  "task_id": 10,
  "tasklist_title": "UI Components",
  "tasklist_description": "Work on header and footer",
  "priority": "high",
  "label_color": "blue",
  "start_date": "2025-08-30",
  "due_date": "2025-09-05",
  "is_completed": false,
  "assigned_to": 2
}
```

29. update_tasks_lists()

- Endpoint: POST /update_tasks_lists/
- Description: Update the details of a task list

- Authentication: ☒ Required
- Request body:

```
{
  "task_list_id": 5,
  "tasklist_title": "UI Components Updated",
  "tasklist_description": "Work on full layout",
  "priority": "medium",
  "label_color": "green",
  "start_date": "2025-08-31",
  "due_date": "2025-09-10",
  "assigned_to": 3
}
```

30. tasks_lists_delete()

- Endpoint: POST /tasks_lists_delete/
- Description: Delete a specific task list from a task card
- Authentication: ☒ Required
- Request body:

```
{
  "task_list_id": 5
}
```

31. copy_task_list()

- Endpoint: POST /copy_task_list/
- Description: Duplicate an existing task list
- Authentication: ☒ Required
- Request body:

```
{
  "task_list_id": 5 // int (ID of the task list to copy)
}
```

32. move_task_list()

- Endpoint: POST /move_task_list/
- Description: Move a task list to another task card
- Authentication: ☒ Required
- Request body:

```
{  
  
  "task_list_id": 5, // int (ID of the task list to move)  
  
  "new_task_card_id": 12 // int (ID of the board where list will be moved)  
}
```

33. sort_task_lists()

- Endpoint: POST /sort_task_lists/
- Description: Reorder task lists inside a task card
- Authentication: ☒ Required
- Request body:

```
{  
  
  "new_task_card_id": 12  
}
```


34. tasklist_checklist_progress()

- Endpoint: POST /task_list_checklist_progress/
- Description: Add checkboxes in Task list and mark them True/False and track its progress out of 100%
- Authentication: ☒ Required
- Request body:

```
{  
  
  "task_list_id": "1",  
  
  "checklist_items": [  
  
    {"name": "option1", "is_checked": true},  
  
    {"name": "option2", "is_checked": false}  
  ]  
}
```


```
}
```

35. convert_checkbox_to_tasklist()

- Endpoint: POST /convert_checkbox_to_tasklist/
- Description: Convert TaskList's Checkbox's items in to your TaskList
- Authentication:  Required
- Request body:


```
{  
  
  "tasklist_id": 1, // int  
  
  "name": "Photo edit" // name of Checkbox's item  
}
```

36. create_comment()

- Endpoint: POST /create_comment/
- Description: Add a comment to a task or board
- Authentication:  Required
- Request body:

```
{  
  
  "tasklist_id": 5, / int (ID of tasklist)  
  
  "comment_text": "This task needs to be completed soon." // string  
}
```

37. edit_comment()

- Endpoint: POST /edit_comment/
- Description: Edit an existing comment
- Authentication:  Required
- Request body:

```
{  
  
  "comment_id": 3, // int (comment to edit)  
  
  "comment_text": "Updated comment content." // string  
}
```

}

38. delete_comment()

- Endpoint: POST /delete_comment/
- Description: Delete a specific comment
- Authentication: ☒ Required
- Request body:

```
{  
  
  "comment_id": 3 // int (comment to delete)  
  
}
```

39. print_export_share()

- Endpoint: POST /print_export_share/
- Description: Export board tasks/data (JSON/CSV/Excel/PDF) and optionally share with others
- Authentication: ☒ Required
- Request body:

```
{  
  
  "task_id": 10, // int (optional, if specific)  
  
  "format": "JSON" // string ("json"/ "csv"/ "excel"/ "pdf")  
  
}
```

❖ Utils(helper) Functions:

1. generate_otp() - Generate 6-digit random otp
2. activity() - Function for generate log activity of user
3. send_otp_email() - Send otp to user Email logic
4. def get_profile_image() - generate full image path, ex."profile_image":
"http://127.0.0.1:8000/media/profiles/20241206_piesUt9.jpg"
5. get_comments() - function for get comment and his Full name only
6. get_checklist_progress() - Function for calculate CheckBoxes progress by 100%

7. `get_image()` - Function for generating full image path, ex. "image":
 "http://127.0.0.1:8000/media/profiles/20241206_piesU†9.jpg"
8. `validate_media_files()` - Function for Validate file and images