Below are our five fully dressed use-cases with a description of a Model usage within them:

Use Case 1

Use Case Name	Creating a Meeting
Scope	Meeting Management
Level	Subfunction
Primary Actor	Users, Discord bot
Stakeholders and Interests	Many users will want to create meetings
Preconditions	None, other than user having initial meeting time proposal
Success Guarantee	The meeting time does not overlap with other times users assigned to it have blocked out
Main Success Scenario	MODEL: User A creates a meeting for next Tuesday at 1pm to 2pm, using an intuitive scheduling system to enter this date and time. Very quickly, the meeting is created and visible for other users to RSVP to. Then, once the meeting is fully created, User A receives a message confirming the successful creation.
Extensions	Meeting time does not work, the user receives an error alerting them to this.
Special Requirements	Users who are related receive alerts telling them about meeting creation
Technology and Data Variations List	Basic I/O, just a command input and text output
Frequency of Occurrence	Meeting creation can be as frequent as half a dozen times per day per user
Miscellaneous	None

Use Case 2

Use Case Name	RSVP to Meeting
Scope	Meeting Management
Level	Subfunction
Primary Actor	Users, Discord bot
Stakeholders and Interests	Once a meeting is created, users will need to be able to signal interest and availability to attend the meeting
Preconditions	A meeting has been created and other users have time available during the meeting's time scheduled
Success Guarantee	The user RSVPing does not have any other meetings scheduled at that time.
Main Success Scenario	MODEL: User B sees "Important Meeting" scheduled by User A. User B selects the RSVP button, and the backend determines that User B is available, and then records User B in a list of RSVPed users. User A and User B both receive a notification about the successful RSVP.
Extensions	If User B does not in fact have a time availability, their RSVP attempt will fail, and only User B will be alerted of this. The meeting will show that an RSVP failed.
Special Requirements	Meeting creator is kept track of to notify when an RSVP is confirmed. The meeting creator can forward the RSVP button to users or otherwise notify them at will.
Technology and Data Variations List	RSVP should be handled via a button.
Frequency of Occurrence	Meeting RSVPing should happen more frequently than meetings, but as much as a dozen times per day per user.
Miscellaneous	None

Use Case 3

Use Case Name	Rescheduling a meeting
Scope	Meeting management
Level	Subfunction
Primary Actor	Users, Discord bot
Stakeholders and Interests	After a meeting is created, it is possible that important people would be unable to attend at that time, thus the meeting would need to be rescheduled.
Preconditions	A meeting has been created and a new time has been decided.
Success Guarantee	The new meeting time does not overlap with other times users assigned to it have blocked out
Main Success Scenario	MODEL: User C sees a meeting for next Tuesday at 1pm to 2pm. Then, he realizes that he would be unable to make the meeting, comes up with a new time and messages User A. User A receives the message and edits the meeting to reschedule it. Everyone that has RSVPed will get notified, and the new time will replace the old time.
Extensions	New meeting time does not work, the user receives an error alerting them to this.
Special Requirements	Users who are related receive alerts telling them about the meeting rescheduling.
Technology and Data Variations List	Basic I/O, just a command input and text output
Frequency of Occurrence	Can occur as many times as needed
Miscellaneous	None

Use Case 4

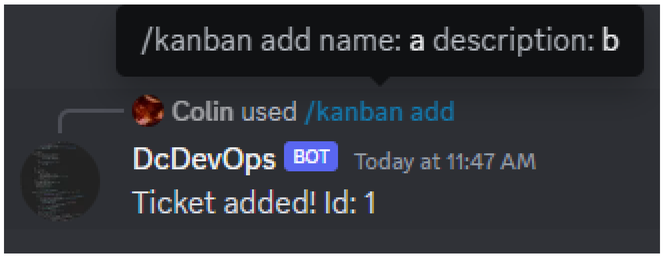
Use Case Name	Creating a ticket
Scope	Ticket Management
Level	Main User Requirement
Primary Actor	Users, Managers
Stakeholders and Interests	Any users and relevant entities will want ticket creation capability in our app.
Preconditions	A task has been determined to be important and has had a name decided to describe it.
Success Guarantee	As long as the command is typed properly, success will occur
Main Success Scenario	MODEL: User A realizes the importance of task “SAMPLE”, then uses the ticket creation command to create task “SAMPLE”. Bot returns that creation was successful.
Extensions	As this task should always succeed, no extensions are clear
Special Requirements	Tickets should be stored in a place that is easily accessible to the user who created it.
Technology and Data Variations List	Input/Output should be handled via a command.
Frequency of Occurrence	Ticket creation will primarily occur during stand up or during meetings with scrum leader, but may occur very frequently during these time periods.
Miscellaneous	Ticket creation is a subtask of ticket management and one of several important ticket interactions.

Use case 5

Use Case Name	Assigning a ticket
Scope	Ticket Management
Level	Main User Requirement
Primary Actor	Users, Managers
Stakeholders and Interests	Any users and relevant entities will want the tickets to be assigned in our app.
Preconditions	The ticket has been created
Success Guarantee	The user assigned the task exists
Main Success Scenario	MODEL: User A knows that user B can do “SAMPLE”, then uses the ticket assign command to assign the task “SAMPLE” to user B.
Extensions	This will always work
Special Requirements	Bot pings User B.
Technology and Data Variations List	Input/Output should be handled via a command.
Frequency of Occurrence	As often as a ticket was created.
Miscellaneous	None

Prototype Showcase

Our prototype Discord bot supports ticket management and meetings. Currently, the prototype is able to add tickets, remove rickets, assign tickets to users, and move them between different statuses. Tickets can be viewed and filtered based on their status and assignee. The prototype also supports scheduling and canceling meetings, but does not yet support RSVPing or checking user availability (though user availability is tracked).

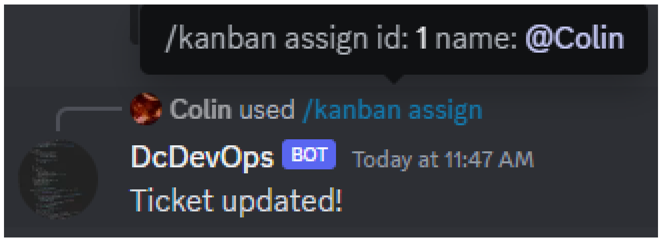


/kanban add name: a description: b

Colin used /kanban add

DcDevOps BOT Today at 11:47 AM

Ticket added! Id: 1

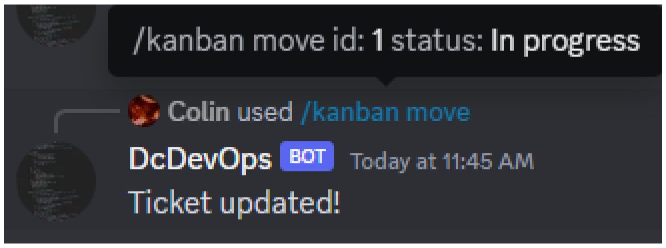


/kanban assign id: 1 name: @Colin

Colin used /kanban assign

DcDevOps BOT Today at 11:47 AM

Ticket updated!

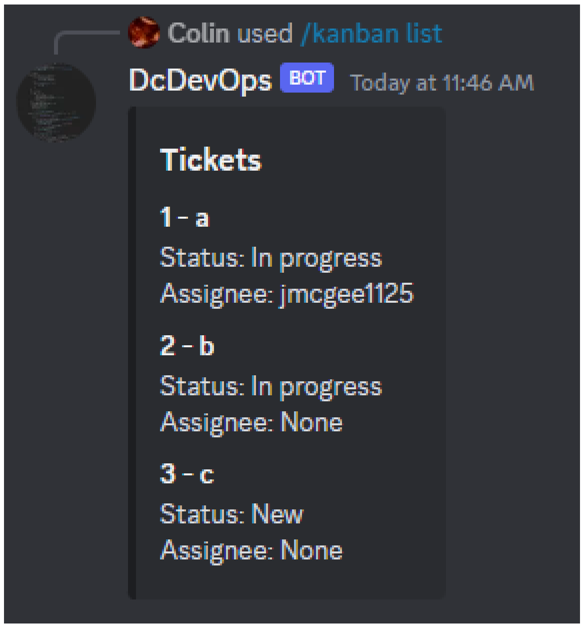


/kanban move id: 1 status: In progress

Colin used /kanban move

DcDevOps BOT Today at 11:45 AM

Ticket updated!



Colin used /kanban list

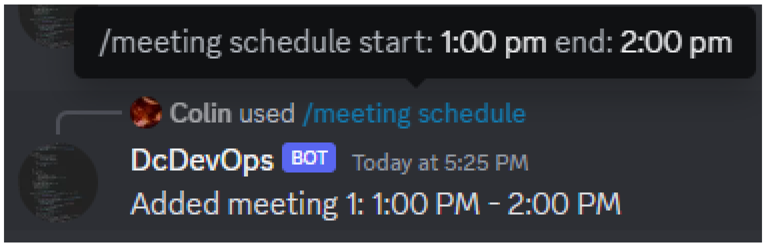
DcDevOps BOT Today at 11:46 AM

Tickets

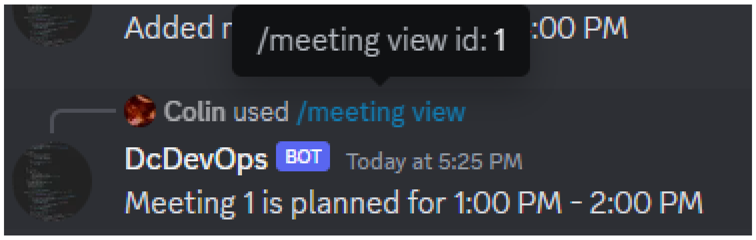
1 - a
Status: In progress
Assignee: jmcgee1125

2 - b
Status: In progress
Assignee: None

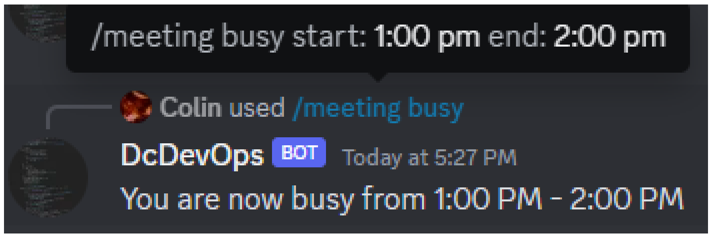
3 - c
Status: New
Assignee: None



```
/meeting schedule start: 1:00 pm end: 2:00 pm
Colin used /meeting schedule
DcDevOps BOT Today at 5:25 PM
Added meeting 1: 1:00 PM - 2:00 PM
```



```
Added r  /meeting view id: 1  :00 PM
Colin used /meeting view
DcDevOps BOT Today at 5:25 PM
Meeting 1 is planned for 1:00 PM - 2:00 PM
```



```
/meeting busy start: 1:00 pm end: 2:00 pm
Colin used /meeting busy
DcDevOps BOT Today at 5:27 PM
You are now busy from 1:00 PM - 2:00 PM
```