Django Admin Command Palette - GSoC 2025 Proposal

proposal.md

Django Admin Command Palette - GSoC 2025 Proposal

Project Information

- **Project Name:** Django Admin: Add Command palette
- **Mentor:** Tom Carrick
- **Difficulty:** Medium
- **Size:** 350 Hours

Project Overview

Summary

This project aims to add a command palette feature to Django Admin, which will improve keyboard navigation and accessibility. The command palette will provide a simple and fast way to search for models, actions, and objects within the Django Admin interface. As someone who has spent a lot of time working with Django Admin, I know how frustrating to switch between the keyboard and mouse. This project will solve that problem by creating a system that allows more keyboard usage, making Django Admin faster and more accessible.

Problem Statement

Django Admin requires a lot of mouse usage for tasks, which makes it difficult for keyboard-only users to navigate easily. When I first started using Django Admin, I noticed how much I had to rely on the mouse. This is a problem for developers who prefer using the keyboard or for users with mobility challenges.

Although the a project `django-admin-keyboard-shortcuts` that provides some shortcuts, it doesn't cover everything. It needs to be expanded into a full command palette, similar to what you find in modern tools like GitHub, VS Code, and JetBrains IDEs, which I use every day to improve my workflow.

Proposed Solution

Design

I propose adding a command palette to Django Admin that work well with the system. When a user presses `Ctrl+K/⌘+K` from anywhere in the admin interface, a modal window will pop up that allows them to:

- Search for and jump to any model admin
- Find and edit specific database objects without clicking through multiple pages
- Perform common actions like save, add, or delete without needing the mouse
- Navigate between sections of the admin interface using keyboard shortcuts
- Customize and extend custom shortcuts for their most common tasks

Also, I will follow some ordered steps to build it which are UI, command system, search system, backend, and extension APIs. I have already created a simple prototype to test this idea. By this solution, users don't require to memorize lots of keyboard shortcuts – they can search for commands directly.

Keyboard Shortcuts

I plan to make The command palette support different keyboard shortcuts to make navigation and action execution faster. Here are the proposed shortcuts:

Global Shortcuts

Name	Key	Description
Open Command Palette	`Ctrl+K / ⌘+K`	Open the command palette from anywhere in the admin
Close Command Palette	`Esc`	Close the command palette
Execute Command	`Enter`	Execute the selected command
Show Shortcuts Help	`?`	Show a dialog with available keyboard shortcuts
Navigate Results	`↑ and ↓`	Navigate through command results
Search for a Model Instance	`Alt + F`	Open search for model instances
Go to Admin Home	`G + I`	Navigate to the admin index page
Show Models List	`G + L`	Show a list of all available models
Go to Model Add Form	`G + A`	Navigate to a model’s add form page
Log Out	`Alt + L`	Log out of the admin interface

Change List Shortcuts

Name	Key	Description
Add New	`Alt + N`	Create a new instance of the current model
Focus Actions Dropdown	`A`	Focus on the actions dropdown
Focus Search	`/`	Focus the search input field
Focus Filter Input	`F`	Move focus to the filter input
Previous Item	`K`	Move to the previous item in the list
Next Item	`J`	Move to the next item in the list
Select Item	`X`	Toggle selection of the current item

Name	Key	Description
Previous Page	`G + P`	Go to previous page of results
Next Page	`G + N`	Go to next page of results

Change Form Shortcuts

Name	Key	Description
Save	`Alt + S`	Save changes and return to list
Save and Add Another	`Alt + A`	Save changes and add a new item
Save and Continue	`Alt + C`	Save changes and continue editing
Delete	`Alt + D`	Delete the current item
View History	`Alt + H`	View history for the current item

Delete Confirmation Shortcuts

Name	Key	Description
Confirm Delete	`Alt + D`	Confirm deletion of item(s)
Cancel Delete	`Alt + C`	Cancel deletion and return

Modified Files

```
django/contrib/admin/
├── templates/admin/
│   ├── base.html                    # Add command palette include
│   ├── change_list.html             # Add page-specific commands
│   └── change_form.html             # Add page-specific commands
├── static/admin/js/
```

```
│   └── core.js                        # Add command palette initialization
└── __init__.py                        # Register default commands
```

Project Timeline

Community Bonding (May 8 – June 1, 2025)

Week 1 (May 8 – May 14)

- Schedule and conduct a 1:1 meeting with the mentor to introduce the project scope and objectives.
- Discuss the proposed architecture and implementation plan.
- Understand existing Django Admin internals relevant to the Command Palette.

Week 2 (May 15 – June 1)

- Finalize the project roadmap and weekly deliverables.
- Refine the technical approach based on mentor feedback.
- Set up the development environment and repository structure.

Phase 1: Core Infrastructure (June 2 – June 22, 2025)

Week 3 (June 2 – June 8): Initial setup and base integration

- Create basic command palette HTML/CSS structure.
- Set up the system to detect keyboard shortcuts.
- Add Ctrl+K/⌘+K as the trigger to open the command palette.

Week 4 (June 9 – June 15): Command registry system

- Develop command registration API.
- Implement context-aware command discovery.
- Build command filtering and search functionality.

Week 5 (June 16 – June 22): Basic UI interactions

- Add keyboard navigation within palette.
- Implement results display and grouping.
- Create shortcut visualization in palette.

Phase 2: Core Functionality (June 23 – July 13, 2025)

Week 6 (June 23 – June 29): Global commands and navigation

- Implement global admin navigation commands.
- Add model listing and navigation functionality.
- Build shortcut help system.

Week 7 (June 30 – July 6): Context-specific commands

- Add change list specific commands.
- Implement change form specific commands.
- Add delete confirmation commands.

Week 8 (July 7 – July 13): Model search functionality

- Create backend API for object search.
- Implement object search UI in palette.
- Build result display and navigation.

Midterm Evaluation Period (July 14 – July 18, 2025)

- Submit progress report and code.
- Review progress with mentor.
- Mentor submits evaluation by July 18, 18:00 UTC.
- Adjust timeline if necessary based on feedback.

Phase 3: Advanced Features and Polish (July 19 – August 10, 2025)

Week 9 (July 19 – July 27): Extension API

- Build developer extension API for custom commands.
- Add documentation for extension system.
- Create example extensions.

Week 10 (July 28 – August 3): Accessibility and visual design

- Implement full keyboard accessibility.
- Add screen reader support.
- Add dark mode support.

Week 11 (August 4 – August 10): Testing and documentation

- Write comprehensive tests.
- Create documentation.
- Prepare for code review.

Phase 4: Final Polishing and Integration (August 11 – August 24, 2025)

Week 12 (August 11 – August 24): Final integration and submission

- Final code review and clean-up.
- Polish UI and ensure consistent design with Django admin.
- Prepare final submission package for GSoC final evaluation.

Final Evaluation Period (August 25 – September 1, 2025)

- Submit final work product by August 25, 18:00 UTC.
- Complete final contributor evaluation.
- Document accomplishments and future work.

- Mentor submits final evaluation by September 8, 18:00 UTC.

Testing

Testing will be an important part of this project to ensure everything works correctly. I will implement **three** main types of tests: **unit tests** to verify individual components and methods, **integration tests** to check how different parts work together, and **cross-browser compatibility tests** to ensure the project works properly across different browsers (including Chrome and Firefox). I will use popular testing tools that match the project's technology stack and maintain a good test coverage. All tests will be well-documented and easy to run, making it simple for other developers to understand and contribute to the project.

Documentation

Documentation will be a key component of this project to ensure long-term usability and maintainability. Good documentation helps both users and developers understand, use, and extend the project. Throughout the development process, I will maintain clear and structured documentation, including setup guides, usage instructions, and contribution guidelines, following the project's standards. It will include code snippets, diagrams, and examples where helpful.

Resources & References

Internal

- Django Documentation
- Django Admin Source Code
- PR #17599
- Ticket #16521

External

- django-admin-keyboard-shortcuts - knyghty
- django-admin-keyboard-shortcuts - arteria
- django-command-palette - rajasimon

- GitHub Command Palette
- VS Code Command Palette
- JetBrains IDE Command Palette
- Notion Keyboard shortcuts
- Notion Command Palette
- Obsidian Command Palette
- Fuzzy Search in JavaScript

About Me

Background

I am a Senior Computer Science Student and Django Backend Developer with a passion for open source contribution and web development. I have hands-on experience building Django applications and contributing to the Django via multiple Pull Requests.

As a Backend Django Developer at Egypt Metro, I implemented a Django backend metro management system which includes API, business logic, and metro administration (Admin Panel) integrating with frontend flutter and AI model. You can view the live application here. Additionally, I have completed a **3 Summer Training** at the Information Technology Institute, mastering full-stack web development technologies such as **HTML, CSS, JavaScript, React, UI/UX, Python, Django, PostgreSQL, and RESTful APIs**. This experience has equipped me with skills in both backend and frontend development.

Contact Information

- Name: Ahmed Nassar
- GitHub: AhmedNassar7
- LinkedIn: nasssar
- Portfolio: Live
- Gmail: a.moh.nassar00@gmail.com
- Django Forum: AhmedNassar7
- Discord: nassar15

- Location: Cairo, Egypt
- Timezone: (UTC+2) Eastern European Time (EET)
- Preferred Language: English