Agendas

Meeting 1 Agenda

Theme: Week 4

Date: 28/8/2022 Time: 3pm Location: Online

Topic 1: Quiz

- Does anyone need help for the weekly quiz?

Topic 2: Work habits

- When does everyone work?
- How does everyone's work progress graph look like for most projects?
- How much does everyone intend to contribute? (Realistically)

Topic 3: Overall Project Direction

- Morph or Evolve (probably not Morph)
- Target User Profile (What audience to cater to?)
- Value Proposition (How can this solve the issue the audience has/improve their lives?)

(**Warning:** DO NOT DISCUSS THE *ACTUAL* PRODUCT/CODE)
[unless it's actually needed for the discussion]

Meeting 1 Minutes

Product Name: TaskBook

Topic 1: Quiz

- No

Topic 2: Work habits

- Equal contribution!

Topic 3: Overall project direction

- Evolve

- Product Name: TaskBook
- User profile: NUS Student
    - Specific profile: John is a leader of his CS2103T group and his CCA (Computing Club), and a member of a group in CS2101 and another CCA (NUS Hackers). He is a bit scatterbrained and has trouble keeping track of what task is assigned to him, and what tasks he assigned to whom. John is a fast typist.
    - Condensed: John is a scatterbrained NUS student who is a fast typist and wants to more efficiently manage his contacts and tasks.
- Value Proposition:
    - This product is an address book that allows for organization of contacts into groups.
    - Record tasks he/she has assigned to a contact in the address book.
    - Record down a task assigned to them by another contact.
    - Allows for better task management so that users can accurately manage events and deadlines assigned to them, or tasks they have delegated.
    - CLI allows for faster access to contacts and tasks than GUI.

- **Direction 1: Evolve** AB3 into a more powerful or more optimized contact management app.
  - Some examples:
    - Manage more entity types related to contacts e.g. Tasks allocated to contacts
    - Contact managing optimized for a specific type of user e.g. a sales person managing client contacts
    - Make existing features *really* strong, worthy of a real product e.g., make the commands more natural, make the search feature more powerful, tweak the GUI to be more useful, ...

  - **pros:** less changes to existing code at the start of the project (i.e., progress will be smoother at the start), can result in a more mature product with deeper features as the product functionality will be moving forward from the start
  - **cons:** less flexibility in product design.

- **Direction 2: Morph** AB3 any direction you wish.
  - For example, an app to manage one of these:
    - Bookmarks of websites
    - Tasks/Schedule
    - Location info
    - Thing to memorize i.e. flash cards, trivia
    - Forum posts, news feeds, Social media feeds
    - Online projects or issue trackers that the user is interested in
    - Emails, possibly from different accounts
    - Multiple types of related things e.g. Contacts and Tasks (if Tasks are allocated to Contacts)
    - ...

  - **pros:** more flexibility in the project direction
  - **cons:** more changes to the existing code at the start while you are still not very familiar with the code base, morphing is less common in real projects (compared to direction 1)

Meeting 2 Agenda

Theme: Week 5

Date: 04/09/2022 Time: 3pm Location: Online

Topic 1: Quiz

- Does anyone need help for the weekly quiz?

Topic 2: Where to do this on

- What medium to record user stories in?
    - Github (Must make the repo)
    - Google Sheets
    - Link:

Topic 3: Brainstorm User Stories

1. How many? (Aim for >30, including obvious ones and look for non-obvious ones)
2. What Version? (Cover until the final version, and include user stories from AB3)
3. What Size? (Can be ‘epic’, but try to make it small)
4. What Format? (Table, or the ‘As a {user type/role} I can {function} so that {benefit}’ format)

Steps:

1. Define persona
2. Define project scope
3. List scenarios to form a narrative
4. Create user stories that fit the narrative (Even nonsensical ones are fine, this is brainstorming).

Topic 4: Submission of User Stories

- Record intermediate steps (personas, scenarios, etc)
- Record brainstormed user stories in online tool discussed in <Topic 2>

Topic 5: Choose User Stories (If we have time for more discussion/For the entire week leading up to the tutorial)

- What user stories to choose for the first iteration of our product?

Meeting 2 Minutes

Narrative 1:

A. First Use (Fresh)
    1. John gets to know about TaskBook. He downloads it and launches it to check out what it can do. (Does he need to install anything beforehand, i.e. Java, etc.?)
    2. John views the available resources to check if TaskBook has all the features he would need for his purposes.
    3. John adds his contacts into Taskbook, he finds that he likes the CLI format of TaskBook.
    4. John tries out the Tasks feature, he likes how he can (organize his thoughts..?)
    5. John quits TaskBook, he likes how TaskBook automatically saves his progress at every stage, so he does not need to worry about accidentally closing without saving.

B. Second Use (New user)
    1. John launches TaskBook. He wants to add his project teammates and record down the tasks he has assigned them.
    2. John wants to see a list of all the tasks he has assigned so far which are still incomplete.
    3. He then begins to record tasks others have assigned to him.
    4. He then too also wants a list of who assigned what task to him.

C. Tenth Use (Familiar user)
    1. John wants to search for the tasks he assigned to his CCA member Tom. He uses the command line to quickly and efficiently get TaskBook to show him all the tasks assigned to Tom.
    2. John has added quite a few tasks. He wants to be able to use the search feature to find tasks based on keywords.
    3. John thinks his tasks are quite disorganized. He wants to be able to sort his tasks by some order other than chronological.
    4. John wants to fine tune the ordering of his tasks.
    5. John accidentally deletes a contact that he did not mean to. He uses the undo command to restore his contact.
    6. John accidentally typos a long and wrong command. He easily navigates to the previous wrong command and edits it.

D. Hundredth Use (Expert user)
    1. John wants more advanced features to optimize his workflow.
    2. He realizes he has too many tasks, and wishes to tag some of them so that he can more easily locate a particular type of task.
    3. He wants to be able to mark and unmark multiple tasks with a single command.
    4. He wants to be able to add and delete multiple tasks with a single command.
    5. He thinks that TaskBook is a really good task management program, so he wants to recommend the program to his friends that are not as good at using CLI..

Uncategorized

AB3-specific

1. As a user, I can add contacts with a single command so I do not have to go through a troublesome GUI.
2. As a user, I can view all my contacts in an alphabetical order if I cannot remember who I'm searching for.
3. As a user, I can delete contacts with a single command so I can easily delete a contact.
4. As a user, I can search the address book with keywords so I can easily find a specific contact.
5. As a user with many contacts, I can add tags to group contacts so I can categorize my contacts.
6. As a user with many contacts, I can search within contact groups, so that I can find related contacts easily.
7. As a user, I can add my contact's address so I do not have to remember the contact's address.
8. As a user, I can add my contact's email so I do not have to remember the contact's email.
9. As a user, I can add my contact's phone number so I do not have to remember the contact's phone number.
10. As a user, I can edit my contacts' information so that I can keep their information up to date.
11. (As a user, I can view my contacts in different sorting orders i.e. name vs phone number, so that I can view my contacts in the most efficient manner)

Taskbook-specific

1. As a new user, I can view a tutorial for how to use TaskBook so that I can know the basics of how to use TaskBook and get started quickly.
2. As a new user, I can use a help command so that I can use TaskBook if I forget the commands.
3. As a user, I can add a deadline to a task, so that I can record when a task needs to be completed by.

4. As a user, I can see the pending task that has the next earliest deadline, so that I can know what I need to do next.
5. As a forgetful user, Taskbook automatically saves my progress after every command so that I do not have to worry about accidentally closing the program and losing my changes.
6. As an adept user, I can view all tasks I have assigned to my contacts with a single command so that I can easily see what others are supposed to do.
7. As an adept user, I can view all tasks I have been assigned by others with a single command so that I can easily see what I need to do for others.
8. As an adept user, I can sort the order of tasks by size, date, etc, so that I can view all tasks with whichever priority is most important to me at any time.
9. As an adept user, I can manually create a custom list order of tasks and/or contacts so that I can fine-tune a task priority for myself.
10. As a careless user, I can undo commands that I have keyed in wrongly so that I can efficiently correct any blunders if I notice them immediately.
11. As a careless user, I can navigate to a previous erroneous command and edit it before sending it again.
12. As an expert user, I can tag tasks so that the tasks can be easily categorized and found using the search keyword feature.
13. As an expert user, I can create shortcuts for tasks so that I can save time on creating frequent tasks.
14. As a potential user that is not familiar with CLI, I can use the secondary GUI components of TaskBook to enter my commands instead, so that I can use TaskBook effectively without the need for being fast at typing.
15. As a forgetful user, Taskbook can detect duplicate tasks so that I would not accidentally add the same task again.
16. As a forgetful user, I can navigate through my command history to look at previous commands.
17. As an expert user, I can redo commands to repeat previous commands so that I can quickly perform bulk actions.
18. As a lazy user, TaskBook can handle my tasks and contacts in one place.
19. As a user, I can delete a task so that a list of tasks is not too long.

- Priority: how important the user story is
- Size: the estimated effort to implement the user story
- Urgency: how soon the feature is needed

User Stories: https://github.com/orgs/AY2223S1-CS2103T-T13-4/projects/1/views/2

PR name: [Team ID] Product Name e.g., [CS2103-T13-4] TaskBook.
PR description: TaskBook is an address book that allows for organization of contacts into groups. It is optimized for CLI users to have faster access to contacts and tasks.

User Guide Distribution of Workload

As a group, please discuss the distribution of the workload for the User Guide. We will be awarding a team mark so you should agree on who does what by completing the table below.

If any member does not contribute based on what is agreed, please inform your tutors and penalties will be applied. However, if you should agree, for example, that one member does all the work, a team mark will still be awarded to all the remaining members. Please discuss your expectations with your team.

Name of Team Member	**Agreed Distribution of Workload**	**Actual Contribution**
Jin Wei	Update quick start. Add instructions for help, contact edit, task edit, undo, redo commands and command history navigation.	As agreed.
Dexter	Update instruction for contact add command. Add instructions for task mark and task unmark commands. Add instructions for task todo, task deadline and task event commands.	As agreed.
Humphrey	Add instructions for adding tasks.	As agreed.
Kevin	Add front matters: - Introduction - GUI breakdown - Useful notations Add the following feature descriptions: - task sort - task find - contact sort - contact find	As agreed.

Bill	Update instructions for Task creation commands, self-assignment for Tasks. Update Quick Start with commands that are valid with sample data.	As agreed.

Note:

The agreed distribution can be adjusted throughout the semester when you have greater clarity of your CS2103T/CS2113T project.

Meeting 3 Agenda

Theme: Week 6

Date: 11/9/2022 Time: 3pm Location: Online

Topic 1: Quiz

- Does anyone need help for the weekly quiz?

Topic 2: Feature List

- Conceptualize product in terms of how it will look like in v1.2

Topic 3: **Draft UG (Submission on LumiNUS)**

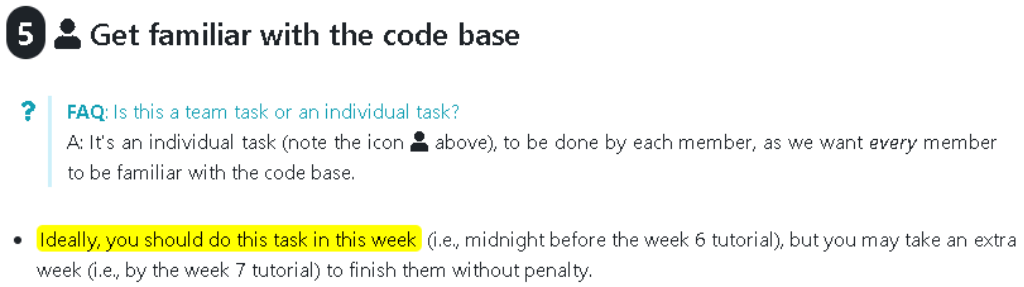
- Start UG Google Doc to describe what the product would be like when it is at v1.2
- Very rough draft, don't waste time formatting, copy editing, etc.
- Just meant to give the tutor a rough idea of the features for this draft.
- Consider UI mock-ups, hand drawn or on sth like PowerPoint.
- Review the draft to ensure features written by each member fit together to form a cohesive product.

Topic 4: Individual Fork Setup

- Follow part 4 of the tP portion

1. "Watch" our tp repo
2. Fork our tp repo to your own personal github account
3. Clone the fork you made into your own computer
4. Follow this: https://nus-cs2103-ay2223s1.github.io/tp/SettingUp.html

Reminder: There is a tP task for us per individual



5 Get familiar with the code base

? **FAQ**: Is this a team task or an individual task?
A: It's an individual task (note the icon above), to be done by each member, as we want *every* member to be familiar with the code base.

- Ideally, you should do this task in this week (i.e., midnight before the week 6 tutorial), but you may take an extra week (i.e., by the week 7 tutorial) to finish them without penalty.

Ideally: midnight before week 6 tutorial (hard deadline: week 7 tutorial)

Meeting 3 Minutes

Feature List (v1.2)

Name	**Remarks**
Add Contact	
Delete Contact	
View Contacts	In alphabetical order.
View all tasks assigned to my contacts	
View all tasks assigned to me	
Add task	
Delete task	
Bye command	To safely exit the application.

Meeting 4 + 5 Agenda

Theme: Week 7

Date: 18/09/2022 + 23/09/2020 Time: 3pm Location: Online

Topic 1: Quiz

- Does anyone need help for the weekly quiz?

Topic 2: Smoke Testing for iP

- Anyone need smoke testing on a particular OS?

Topic 3: Practice Iteration

- Issues
- Milestones
- Set the milestone up before continuing

Topic 4: Project Website

- Who updates what section of what document?
- Documents to update/start doing:
    - AboutUs
    - README

Topic 5: Update the UG

- Move UG to GitHub Pages
- Who does which part?

Topic 6: Skeletal PPP

- Just headings for now
- Follow instructions

Topic 7: Update the DG

- Who updates what section?
- Target user profile, value proposition, user stories
- Use cases

- NFRequirements
- Glossary

Topic 8: Plan Next Iteration (23/09/2022)

- Any tweaks?
- Divide work
- Add said work to issue tracker and appropriate milestone

If you have time, start implementing v1.2 after finishing v1.1.
Note: v1.2 is for absolute requirements that are needed for an MVP. We can then add important but not essential features that improve upon v1.2 features in v1.2b, then all other features we want to have in v1.3.

Meeting 4 Minutes

Roles and responsibilities:

- ***Kevin*** *Team lead*: Responsible for overall project coordination.
- *Documentation* (short for 'in charge of documentation'): Responsible for the quality of various project documents.
- ***Bill*** *Testing*: Ensures the testing of the project is done properly and on time.
- ***Dexter*** - *Code quality*: Looks after code quality, ensures adherence to coding standards, etc.
- ***Humphrey*** *Deliverables and deadlines*: Ensure project deliverables are done on time and in the right format.
- ***Jin Wei*** *Integration*: In charge of versioning of the code, maintaining the code repository, integrating various parts of the software to create a whole.
- ***Dexter/Kevin\ 200% -*** *Scheduling and tracking*: In charge of defining, assigning, and tracking project tasks.
- **Kevin** Gradle Expert & Agenda Maker
- *[Tool ABC] expert*: e.g. Intellij expert, Git expert, etc. Helps other team member with matters related to the specific tool.
- *In charge of[Component XYZ]*: e.g. In charge of `Model`, `UI`, `Storage`, etc. Note that being *in charge* of a component doesn't mean you are the only one who should be modifying that component. Rather, you are the one who's expected to, know that component best, review changes done to that component in v1.3-v1.4, act as the

> gate keeper of its quality, help others when they face difficulties while modifying that component etc

AboutUs

For “in charge of component”, we split such that each member is mainly in charge of one component and is also sub-in-charge of another component to overlap.

✅Kevin: Team Lead, Gradle Expert, Agenda Maker, In Charge of UI, Logic
✅Bill: Testing, In Charge of Model, Storage
✅Dexter: Code Quality, Scheduling and Tracking, In Charge of Logic, Storage
✅Humphrey: Deliverables and Deadlines, In Charge of Model, UI
✅ Jin Wei: Integration, In Charge of Logic, Commons

Everybody makes an issue, makes a PR for their own update.

README

(root/readme.md)

1. ✅**Kevin** UI Mockup
2. ✅**Bill** Update contents to match TaskBook
3. ✅**Bill** Paste in README: This project is based on the AddressBook-Level3 project created by the [SE-EDU initiative](https://se-education.org).
4. ✅ **Jin Wei** Fix Jekyll site-wide settings

Update the UG

(docs/userguide.md)

1. Each person transfer from our pdf to repo
2. Put [coming soon!] for unimplemented features

Transfer to UserGuide.md
Everyone: Command summary
✅Kevin: View Tasks [both user -> contact and contact -> user], View contacts
✅Bill: Add Deadline, Date formatting
✅Dexter: Add contact, phone number, email, address
✅ Humphrey: Add task, FAQ, Command Summary
✅ Jin Wei: Delete Contact, Delete Task, Quick Start

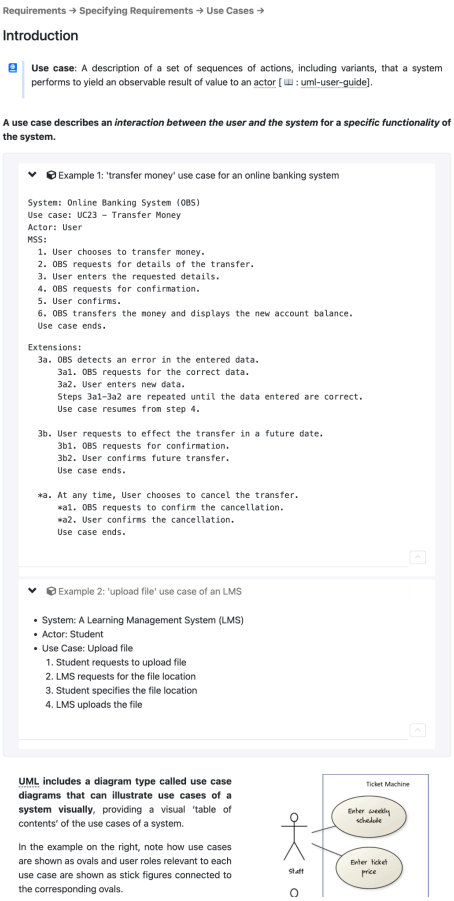
PPP

`docs/team/github_username_in_lower_case.md`
Everyone does themselves in their own file.
Checklist: ✅Kevin, ✅Bill, Dexter, ✅Humphrey, ✅Jin Wei
Everybody make an issue, make a PR

Developer Guide

Follow this: https://se-education.org/addressbook-level3/DeveloperGuide.html
DeveloperGuide.md

- ✅Kevin - Target user profile, value proposition, and user stories: Update the target user profile and value proposition to match the project direction you have selected. Give a list of the user stories (and update/delete existing ones, if applicable), including priorities. This can include user stories considered but will not be included in the final product. (Done, no PR yet. The changes are currently temporarily on Discord)
- Dexter Humphrey - Use cases: Give use cases (textual form) for a few representative user stories that need multiple steps to complete. e.g. Adding a tag to a person (assume the user needs to find the person first)



Requirements → Specifying Requirements → Use Cases →

Introduction

Use case: A description of a set of sequences of actions, including variants, that a system performs to yield an observable result of value to an actor [: uml-user-guide].

A use case describes an *interaction between the user and the system* for a *specific functionality* of the system.

Example 1: 'transfer money' use case for an online banking system

```
System: Online Banking System (OBS)
Use case: UC23 - Transfer Money
Actor: User
MSS:
  1. User chooses to transfer money.
  2. OBS requests for details of the transfer.
  3. User enters the requested details.
  4. OBS requests for confirmation.
  5. User confirms.
  6. OBS transfers the money and displays the new account balance.
  Use case ends.

Extensions:
  3a. OBS detects an error in the entered data.
      3a1. OBS requests for the correct data.
      3a2. User enters new data.
      Steps 3a1-3a2 are repeated until the data entered are correct.
      Use case resumes from step 4.

  3b. User requests to effect the transfer in a future date.
      3b1. OBS requests for confirmation.
      3b2. User confirms future transfer.
      Use case ends.

  *a. At any time, User chooses to cancel the transfer.
      *a1. OBS requests to confirm the cancellation.
      *a2. User confirms the cancellation.
      Use case ends.
```

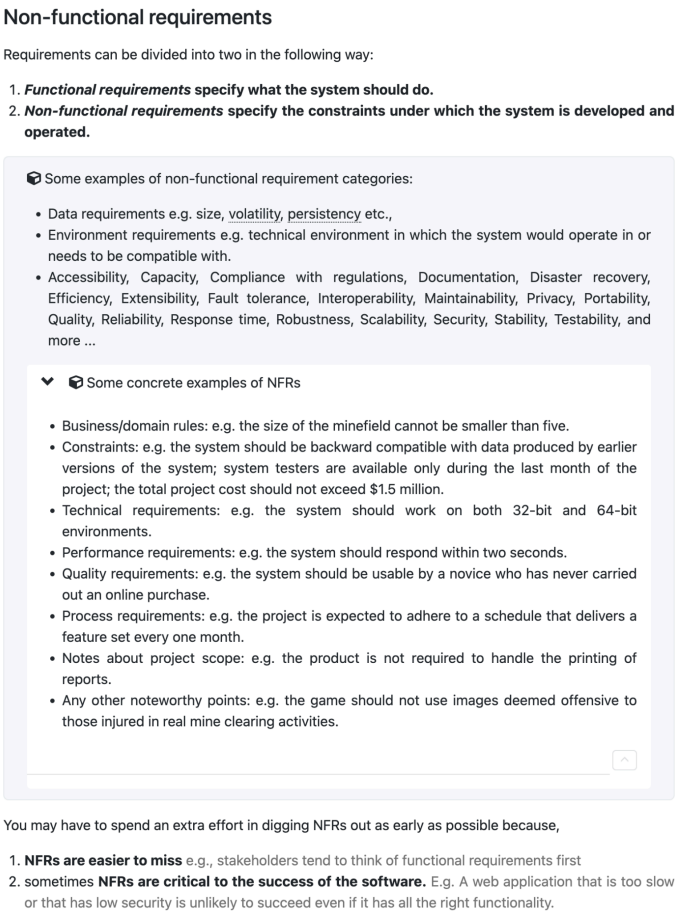
Example 2: 'upload file' use case of an LMS

- System: A Learning Management System (LMS)
- Actor: Student
- Use Case: Upload file
  1. Student requests to upload file
  2. LMS requests for the file location
  3. Student specifies the file location
  4. LMS uploads the file

UML includes a diagram type called use case diagrams that can illustrate use cases of a system visually, providing a visual 'table of contents' of the use cases of a system.

In the example on the right, note how use cases are shown as ovals and user roles relevant to each use case are shown as stick figures connected to the corresponding ovals.

- Non-functional requirements: Note: Many of the given project constraints can be considered NFRs. You can add more. e.g. performance requirements, usability requirements, scalability requirements, etc. - Bill, Jin Wei
- https://nus-cs2103-ay2223s1.github.io/website/admin/tp-constraints.html



Non-functional requirements

Requirements can be divided into two in the following way:

1. ***Functional requirements*** **specify what the system should do.**
2. ***Non-functional requirements*** **specify the constraints under which the system is developed and operated.**

Some examples of non-functional requirement categories:

- Data requirements e.g. size, volatility, persistency etc.,
- Environment requirements e.g. technical environment in which the system would operate in or needs to be compatible with.
- Accessibility, Capacity, Compliance with regulations, Documentation, Disaster recovery, Efficiency, Extensibility, Fault tolerance, Interoperability, Maintainability, Privacy, Portability, Quality, Reliability, Response time, Robustness, Scalability, Security, Stability, Testability, and more ...

Some concrete examples of NFRs

- Business/domain rules: e.g. the size of the minefield cannot be smaller than five.
- Constraints: e.g. the system should be backward compatible with data produced by earlier versions of the system; system testers are available only during the last month of the project; the total project cost should not exceed $1.5 million.
- Technical requirements: e.g. the system should work on both 32-bit and 64-bit environments.
- Performance requirements: e.g. the system should respond within two seconds.
- Quality requirements: e.g. the system should be usable by a novice who has never carried out an online purchase.
- Process requirements: e.g. the project is expected to adhere to a schedule that delivers a feature set every one month.
- Notes about project scope: e.g. the product is not required to handle the printing of reports.
- Any other noteworthy points: e.g. the game should not use images deemed offensive to those injured in real mine clearing activities.

You may have to spend an extra effort in digging NFRs out as early as possible because,

1. **NFRs are easier to miss** e.g., stakeholders tend to think of functional requirements first
2. sometimes **NFRs are critical to the success of the software.** E.g. A web application that is too slow or that has low security is unlikely to succeed even if it has all the right functionality.

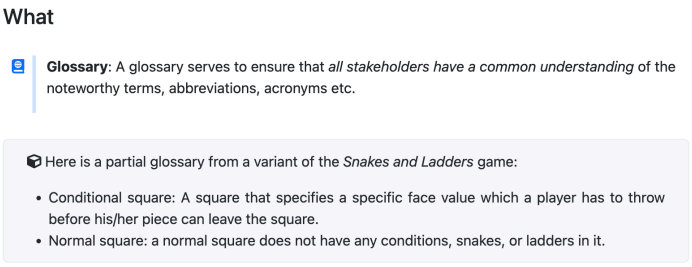
Our NFRs

(first 3 already in the document, red is edit)

1. Should work on any mainstream OS, on 32-bit or 64-bit environments, as long as it has Java 11 or above installed.
2. Should be able to hold up to 1000 persons and 5000 tasks without a noticeable sluggishness in performance for typical usage. The product should be responsive to any command within 2 seconds.
3. A user with above average typing speed for regular English text (i.e. not code, not system admin commands) should be able to accomplish most of the tasks faster using commands than using the mouse.
4. The system's stored data should be backwards compatible with all previous versions.
5. The system's data storage file should be human-editable.

6. The system should l~~ose minimal to no data~~ on unexpected crashes.
   The product should not cause data corruption on unexpected crashes
7. The system should be reliable and work as expected most of the time.
8. The user interface should be intuitive enough to be usable by a novice with little experience but works more efficiently for an experienced user.
~~9. The source code should be open source.~~
10. The product is offered as a free downloadable offline application without requiring the use of an installer.
11. The product should work offline.
12. T~~he product is used by a single user.~~
    The product should work with a standalone user.
13. The product's GUI should work well for most standard screen resolutions or higher.
14. The product should be packaged into a single JAR file with a maximum size of 100MB.
15. The product's data should be portable to allow the user to export their data.
16. The product should be easy to test, with appropriate manual testing instructions and automated tests.
~~17. The system should be built to be scalable with other features.~~

- Glossary: Define terms that are worth recording. - Jin Wei



Docs we need to update (eventually):

- should we delete johndoe.png & johndoe.md (ppp)

v1.2 Implementation

Kevin: Ui for viewing tasks right of contacts & list tasks command (implement with ability to extend to view user/others tasks)
Bill: Task model
Dexter: TaskBookParser -> parser for commands
Humphrey: command executioner
Jin Wei: TaskBookParser -> parser to call various parsers (by tmr also, hopefully)

(and we need to do the actual task command functionalities after these ^)

Task:

- Name
- Description
- Flag: assignee / assignor
- Assignee / Assignor
- Done

TaskList

Meeting 6 Agenda

Theme: Week 8

Date: 02/10/2022 Time: 3pm Location: Online

Topic 1: Quiz

- Does anyone need help for the weekly quiz?

Topic 2: Issue Tracker proper setup

- More accurately, check that all our current issues are properly tagged with correct labels

Topic 3: tP expectations

That's about it, we (or rather, everyone except Dexter and Kevin) have midterms in the upcoming week, so this meeting will be relatively ***fast***.

Remember to evaluate your 2 assigned iPs by 8th October! (Saturday 2359)

Also, do add some forms of testing for your code. Then assign the PR with the new test to Bill(?) for him to check. ok.

Meeting 6 Minutes

- We should add a deadline to our v1.2 milestone in github as per the tp progress tracker.

Bill - Model Manager, Unit tests for base models, Integration tests(?)
ObservableTaskList: Model(Model Manager) -> logic -> UI
Dexter - unit tests
Humphrey - execute methods for addtask,deletetask,tasklist
Kevin - unit tests for UI, DG, try to improve the look of the UI more (maybe make the windows display vertically)

Jin Wei - get model manager from logic to UI (in logic manager), Update UML diagrams

Main changes try to finish by: next meeting (aka Sunday)
Soft deadline: next Tuesday-ish
Hard deadline: next Thursday

Meeting 7 Agenda

Theme: Week 9

Date: 09/10/2022 Time: 3pm Location: Online

Topic 1: Quiz

- Does anyone need help for the weekly quiz?

Topic 2: v1.2 checkup

- Are there any outstanding issues not done?
- How close are we to closing the v1.2 milestone?
- If all done, close it and tag the current commit as v1.2
- Do a release on GitHub, but .jar file is optional.

Topic 3: Informal demo (This is probably going to be the main thing we will be doing this week)

- Screen record or Screenshots
- Decision: Kevin will do.

Reminders:
TEAMMATES survey due tonight.
Tag the commit where we close the milestone with v1.2
Figure out how to test code coverage on IDE (https://www.jetbrains.com/help/idea/running-test-with-coverage.html)
Prepare the tutorial diagrams (OODMs, Activity Diagrams)
Take a good nap.

Soft deadline for v1.2 is TUESDAY (refer to Meeting 6 minutes)! Try to make sure everything is done by then at the latest!

Meeting 7 Minutes

On Tuesday:

1. Tag on GitHub
2. Make release on GitHub

Meeting 8 Agenda

Theme: Week 10

Date: 16/10/2022 Time: 3pm Location: Online

Topic 1: Quiz

- Does anyone need help for the weekly quiz?

Topic 2: v1.2 Postmortem (SUBMITTABLE)

- What worked well?
- What didn't work?
- How to improve the process? (NOT THE PRODUCT)

Topic 3: Adjust process rigor if needed

- Do we need this?

Topic 4: Add user stories to v1.3

- More specifically, assign the ones we are doing to v1.3
- This is the last iteration we can add features in!

Topic 5: Update DG (**MUST BE DONE BEFORE DEADLINE**)

- Deadline is Thursday (I think)
- Refer to site for details
- All to do by Tuesday/Wednesday (TBD in meeting)

Topic 6: Smoke Test CATcher (**MUST BE DONE BEFORE DEADLINE**)

- Deadline is Thursday (I think)

Topic 7: JAR release (**MUST BE DONE BEFORE DEADLINE**)

- v1.2.1 or v1.3 trial

Tutorial prep:
Demonstrate assertion failure screenshot
Defensive programming examples in our tP (MUST ADD)

Demo and Pitch plans to be done by Tuesday

Meeting 8 Minutes

Post-Mortem

What worked well?

- Defined the blockers early so that we can focus on them
- Everyone has ownership of their parts
- Everyone completed their stuff on time

What didn't work well?

- Difficult to maintain integration tests as they require dependencies from multiple components and thus multiple people
- Communication is occasionally inefficient across components because of a lack of understanding of that component
- Large PRs are difficult to understand and review

How to improve process:

- Take part in PR reviews together
- Better explain the components we are in charge of to our teammates
- Add more descriptive explanation about changes made in PR
- Break larger PRs into a few smaller ones
- When a component is changed, briefly explain the changes made and why they are needed in the communication channel.

Adjust process rigor

- Branching workflow from new branches.

Developer Guide

At least one enhancement you added.
Needs to be 1+ pages long.
The description can contain things such as,

- How the feature is implemented (or is going to be implemented).
- Why it is implemented that way.
- Alternatives considered.

Bill: Storage
Dexter: Command and Parser
Humphrey: Task commands execute & tags
Kevin: SortedList and UI
Jin Wei: Command history navigation and Undo [planning to add]

JAR Release
Wait for proper storing of new task types.
Release tomorrow (9pm Monday).

Meeting 9 Agenda

Theme: Week 11

Date: 23/10/2022 Time: 3pm Location: Online

Topic 1: Quiz

- Does anyone need help for the weekly quiz?

Topic 2: Update User Docs

- Landing page

Topic 3: Release as a jar file

- Soft deadline: Tuesday midnight
- **Hard deadline: Friday 10am**
- Note: Can do second release after if last min update before dry run.

Topic 4: Wrap up v1.3 (after releasing as jar file)

- Git tag, close all issues we can.
- Do release

Topic 5: Demo v1.3

- As before, need not be polished

UGDG draft for 2101
PLS send Kevin any GUI format suggestions

Meeting 9 Minutes

- Update the v1.3 user guide to match the current version of the product. Reason: testers will need to refer to the UG during the practical exam dry run.
    - Clearly indicate which features are not implemented yet e.g. tag those features with a `Coming soon`.

- For those features already implemented, ensure their descriptions match the exact behavior of the product e.g. replace mockups with actual screenshots

Update UI picture, update landing page! (must do by this week)

Final Features
Kevin - Specific find for task/contact, add message after filter/sort. UI final formatting.
Bill - Self person, Storage for tagging of tasks, Manual save.
Humphrey - List all tasks -> Reset filter/sorted, Tagging of tasks.
JW - Help command -> generic help + individual command help., do task edit tags after humphrey tags the tasks
Dexter - Optional Contact fields - if no prefix -> set as default “No phone number/address/etc.”

Tags - UI, Find, Filter, Edit - Same as Contact (Reset the entire list if want to edit)

-> update Ui.png (maybe we can do while we are releasing v1.3)

Meeting 10 Agenda

Theme: Week 12

Date: 30/10/2022 Time: 3pm Location: Online

Topic 1: PE-D Bug Analysis

- Let's go through all the issues that the testers have found
- Which ones are legit, which ones are nitpicky, which ones are useful, which ones are spurious (*definition: not being what it purports to be; false or fake*)?
- Tester A: https://github.com/Jo3LW/ped/issues (10)
- Tester B: https://github.com/dr-arrgghh/ped/issues (7)
- Tester C: https://github.com/johnbenedictyan/ped/issues (9)
- Tester D: https://github.com/prit3010/ped/issues (6)
- Tester E: https://github.com/ruihan00/ped/issues (7)

PE-D issues received: 39

Unique bugs: 34

- Not allowed to fix in v1.4: 1
- Must fix: 13 (fixed: 9, pr in review: 1)
- Good to fix: 18 (fixed: 15, pr in review: 2)
- Won't fix or invalid: 2

Topic 2: UGDG

- Need to try to get UG done TODAY! (Done)
- UG draft consultation tmrw!
- The closer to completion our draft is, the better feedback we will have!
- Fix bugs found in Minutes, compiled by Jin Wei (Done)

Topic 3: Product Pitch and Demo

- Demo is on THURSDAY!
- Get slides (if any) done by Tuesday to ensure enough time to practice/digest!
- Pitch is next Monday

- Should try to get slides done by Friday…?

Topic 4: PPP
- Must be done by the usual deadlines
- Tracked by individual progress tracker

Topic 5: Reposense (OTOT)
- Check if Reposense has correctly tracked your code contribution

Topic 6: PDF conversion
- Try it out

Meeting 10 Minutes

We need to review the testers in Teammates also. DONE.

Feedback provided from CS2101 peer-review session

UG: Table of Contents is broken (indentation)
UG: purpose & target audience
UG: instructions for how to use the guide (how to navigate/icons interpretation)
UG: merge notes about command format + accepted date format into one area
UG: can make more use of "note", "info", "important"
UG: consider make "list" a section and then "contact list" & "task list" as subsections (for all our commands)
UG: input flag, sort flag -> inconsistent language
UG: examples for command summary in separate column?
UG: edit -> "all included fields" -> not clear what is "included fields" (could have better wording)
UG: image captions

DG: "3rd party libraries" formatting is incorrect
DG: target reader
DG: a few proposed implementations could have been more comprehensive
DG: update glossary (do we need to?)
DG: break NFR into sub-sections?
DG: new developers section
DG: label diagrams?
DG: sort user stories table?

DG: Grammar error: “as an taskBookStateList:”, Spelling error: “prioriity” , British English: “organised”, “memorise”

Meeting 11 Agenda

Theme: Week 13

Date: 6/11/2022 Time: 3pm Location: Online

Topic 1: UG 5 man review

- Everyone to look through the UG
- Check if all the points raised against us by prof have been addressed

Topic 2: DG 5 man review

- Everyone to look through the DG after DG is done
- Areas left to improve:
    - Glossary
    - Review Use Cases and 1 use case for all main commands
    - Update value proposition to whatever we have
    - Is the architecture diagram we have correct rn

Topic 3: PPP

- Everyone to note: All other deliverables are 1 man per team submit, but the PPP is an individual submission.

Topic 4: Module website

- Everyone just check

Topic 5: Self-smoke test JAR file

- Check all 3
- Next, all to spend at least 30 mins (but try for 1 hour) to break TaskBook.
- Does not have to be during meeting, but to be done asap so that we have time to fix any ultra critical bugs

---

Meeting 11 Minutes

Meeting 12 Agenda

Theme: Week 13 Demo Video

Date: 8/11/2022 Time: 9pm Location: Online

Topic 1: Record video

- Let's begin recording!

Meeting 12 Minutes

Script (18 min limit):

Bill

Welcome to the demo of our product TaskBook, a no-frills desktop application for managing all your tasks.
TaskBook is targeted at all NUS students who want to manage their tasks consistently, but have yet to find a single system to do so. It is also optimized for fast typists as commands are provided through the Command Line Interface (CLI).

[User Interface]
At the top navigation bar, you will see File and Help. The File dropdown allows you to exit the application, whereas the Help dropdown will display a pop up with a link to TaskBook's user guide. You may find detailed explanations of all the features here.

Below the navigation bar, you can find the command box. Here, you will be entering various commands to interact with the application.

Right below that, there is the results display box where you will see the message after each command. This is also where errors are displayed.

Moving on, you have the contact list on the left panel and the task list on the right panel. Here you will see the relevant contacts and tasks depending on your command. By default, all your contacts and tasks will be shown.

[Help]
`help`

Firstly, you can use the help command to view all the available commands in TaskBook.

It also specifies that we can use help c/COMMAND to view details about individual commands.

Here c/ "c backslash" is known as a flag, it consists of an alphabet and backslash and it tells TaskBook which part of the command you are about to enter next.
COMMAND, or any words in capital letters represents a field that the user must provide. In this case, you are able to replace it with any of the commands specified in the list.

—----------

Jin Wei

I will be continuing from the core features.

[Contact Add]
`help c/contact add`

Let's check out the message usage of contact add.

`contact add n/Damith p/61564359 #/Professor`
Let's add Professor Damith as a contact, along with his phone number. As he is our professor, let's add a #/Professor tag. Only the name is a compulsory field. The other fields will be replaced with an indicator when left blank.

Creating contacts is a core feature of TaskBook. A contact can have a name, phone number, email, address and even tags.

Next, let's check out the tasks. Tasks have an assignor assignee feature, along with a description and optional tags. There are 3 different tasks types, which serve different purposes. Let's go through them one by one.

[Task Todo]
`help c/task todo`

Now, let's create a todo task for demonstration. Without specifying the assignor or assignee, the task will default to be assigned to Myself.

`task todo d/This is a demo`

Todos are the most generic form of task, and are used for taking down quick tasks without a deadline associated with them.

[Task Deadline]
`task deadline m/Damith d/Demo Poster t/2022-11-09 #/FinalSubmission`

Next, let's add a deadline task. Professor Damith has assigned us a task: a demo poster for our product, due 9th November 2022. Let's add this task, and set him as our assignor. We can tag this as #/FinalSubmission.

Deadline tasks are todos with deadlines associated with them. They are used when a particular task has to be done before a specific date.

[Task Event]
`task event m/Bernice Yu d/Team Meeting t/2022-11-09`
Next, let's add an event task, assigned by Bernice Yu. A date can be added similarly to a deadline.

Events are used for tasks that represent an event instead of an action.

Notice how the 3 different task types are visually distinct from each other.

Prof just sent us an email, let's save his email for future use!

`contact find q/Admith` - intentionally wrong
However, we have too many contacts and scrolling is tough work, so let's try searching for him first.

`contact list`
Unfortunately, a typo has been made, but no worries as the list of contacts can be easily reset back to the full view again by using contact list command.

[Contact Find]
`contact find q/Dam`
Let's try again, but let's just search for a portion of his name instead. This allows us to quickly find a particular contact from our huge list of contacts.

[Contact Edit]
`contact edit i/1 e/dscdcr@nus.edu.sg`
Now, let's add his email. We enter the displayed index, 1, and the new email to change it.

Editing contacts allows us to both add in new information and modify wrong data.

—----------

Dexter Sim Choon Hong, D.Va / Sage main btw

[Contact Sort]
By default, TaskBook stores the contacts in chronological order.

`help c/contact sort`
If we do not remember the exact contact details to use the find command, the sort command provides various sorting orders that allow us to find contacts more easily.

`contact sort s/a`
We can sort them in alphabetical order by providing a to the s flag.

`contact sort s/ra`
By changing the field in the s flag to ra, we can sort contact in reverse alphabetical order. This allows us to quickly locate contacts that start with the last letters of the alphabet.

`contact sort s/ca`
By changing the field in the s flag to ca, we can sort contacts chronologically by the time they were added into TaskBook. This allows us to find the contacts that we have recently added into TaskBook. As mentioned before, this is the default sorting mechanism used by TaskBook.

[Contact Add]
`contact add n/Matthew Tan #/ComputingClub`
Now, we have to attend the Computing Club's Annual Meeting. The invite was sent out by the Computing Club's president Matthew Tan. We will add a new contact Matthew Tan. However, we don't know any of his contacts details yet, so we will just add a #Computing_Club tag, and the remaining fields will be left empty. Note that tags do not support whitespaces to encourage short tags. We can add his contact details in the future with the contact edit command when we find out his contact details.

[Task Find]
`task find q/Demo Poster`
Prof Damith just told us that he actually wants us to do a demo video instead, so let's edit the deadline task we added earlier. Just like before, we can search for the task using task find.

Unlike contact find, task find supports searching using multiple fields, namely query in description, task association and their completion status. Multiple fields allow TaskBook to more accurately search for the right task, especially when the task list contains a lot more tasks.

[Task Edit]
`task edit i/1 d/Demo Video`
Now, we simply edit the description by calling the task edit command with index 1 and new description Demo Video. This will change the description of the task at specified index and reset the view of the task list to the full list.

[Task Find (but with multiple parameters)]
`task find x/O a/FROM`

To demonstrate that we can find tasks with multiple parameters, we will use the task find command to find all the tasks that are assigned by other people and are not done. The x flag with field O represents tasks that are not marked as done. The a flag with field FROM represents tasks that are assigned by other people.

[Task List]
`task list`
Now, we use the task list command to reset the task list back to its original state.

—----------
Humphrey

[Task Find]
`task find q/Demo Poster`
Let's use the task find command again to check if the Demo Poster task still exists.

Since we've edited the description, the find command now returns an empty list.

[Task List]
`task list`
Finally, we can reset the list of tasks shown back to the full list of tasks with the task list command.

[Task Find]
`task find q/Demo Video`
Let's use the task find command again to check if the task was updated properly. The edited Demo Video task should show up.

Now, let's demonstrate how we can sort the displayed tasks.
`task list`
Let's first reset the list again.

[Task Sort]
By default, TaskBook stores the tasks in the order that we added them in.

`help c/task sort`
If we do not remember the exact task description to use the find command, the sort command provides a few ways to sort our tasks so that we can find the tasks we want more easily.

[Task Sort]
`task sort s/a`
Using the 'a' argument to the 's' flag, you can also sort by alphabetical order based on a task's description…

`task sort s/ra`
Or reverse alphabetical order with the ‘ra’ argument..

`task sort s/cd`
If we want to see which tasks are due soon, we can sort by chronological order based on the date assigned to the task using the ‘cd’ argument..

`task sort s/rcd`
Alternatively we can sort them by reverse chronological order using the ‘rcd’ argument.

`task sort s/ca`
However, if you want to sort based on the time they were added into TaskBook, you can use the ‘ca’ argument instead.

Take note that when sorting based on assigned date, tasks with no assigned dates are placed at the end of the list in no particular order.

TaskBook also supports simultaneous usage of sort and find commands, so you can use them in combination with each other.

[Task Mark]
`task mark i/11` - intentionally wrong
After some time, we are done with Damith’s Demo Video task, so let’s mark it as done with the task mark command.

[Task Unmark]
`task unmark i/11`
Oh no, we’ve marked the wrong task as done! Not to worry, we can simply unmark the task with the task unmark command.

—----------

Kevin

[Task Mark]
`task mark i/12`
Now let’s correctly mark the demo video task.

[Contact Delete]
`contact delete i/11` - intentionally fail
Since we are done with the task, and we are also not going to need Professor Damith in our contact list anymore, let’s delete them.

[Task Delete]

`task delete i/12`
Oh, we can't delete Professor Damith because the demo video task is still in our task list. Let's do that first then.

We prevent users from deleting contacts with tasks to prevent unintentional deletions from the users. Perhaps they thought they were done with all the tasks related to the contact when they actually weren't.

`contact delete i/11` (can use double up arrow key, will be explained at the end)
Now, let's delete Professor Damith from the list.

[Undo]
`undo`
New information just came in, we still have a final practical examination, so we shouldn't have deleted Professor Damith's contact. Not to worry, we can simply undo that previous delete operation. :(

Undo safely stores up to 15 previous states of TaskBook. Any command that affects TaskBook's contacts or tasks is counted as a change in state. Note that this means find and sort commands to not change state.

[Redo]
`redo`
For completeness, we also offer redo: a way to 'undo the undo', in case the user goes back too far by accident using undo.

[Command History Navigation]
`UP` and `DOWN` arrow keys
<scroll through command history, up and down while narrating>
Everything we have just done is stored in the command history. You can use the up and down arrow keys to navigate through them, just like how you would on your typical terminal. This history is not saved between user sessions.

[Bye]
That's all for this demo. TaskBook uses these features to effectively manage the relationships between tasks and contacts, thus making tasks easier to remember.

Finally, you can close the application with the bye command.

Features:

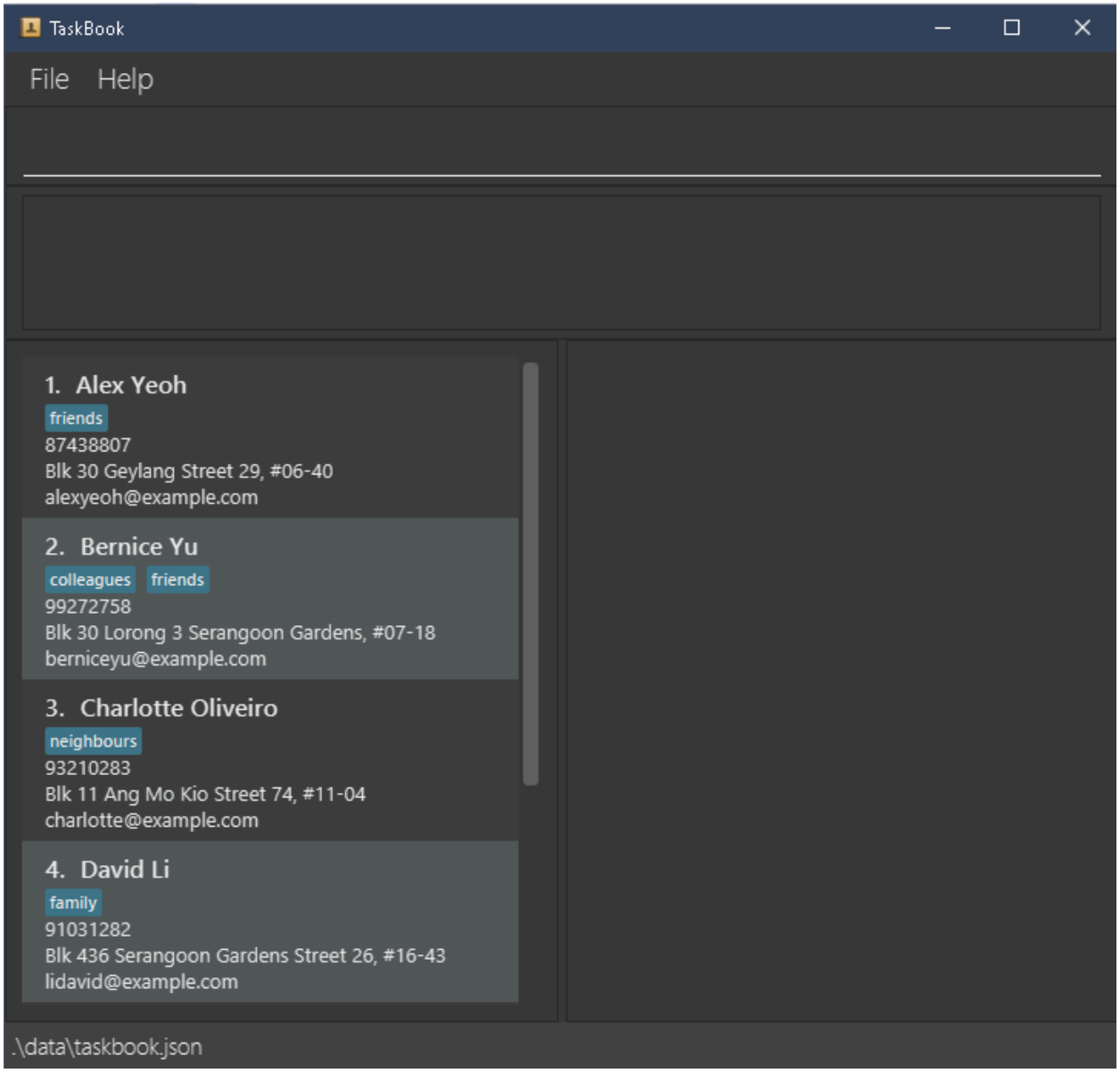
- ✅ Viewing help : `help`
- Listing
    - ✅ Listing all Contacts : `contact list`

    - ✅Listing all Tasks : `task list`
- Adding
    - ✅Adding a contact : `contact add`
    - ✅Adding a todo : `task todo`
    - ✅Adding a deadline : `task deadline`
    - ✅Adding an event : `task event`
- Editing
    - ✅Editing a contact : `contact edit`
    - ✅Editing a task : `task edit`
- Deleting
    - ✅Deleting a contact : `contact delete`
    - ✅Deleting a task : `task delete`
- Finding
    - ✅Finding contacts : `contact find`
    - ✅Finding tasks : `task find`
- Sorting
    - ✅Sorting contacts : `contact sort`
    - ✅Sorting tasks : `task sort`
- Marking as done
    - ✅Marking a task : `task mark`
    - ✅Unmarking a task : `task unmark`
- Undo and Redo
    - ✅Undo Command : `undo`
    - ✅Redo Command : `redo`
- ✅Exiting the program : `bye`
- ✅Navigating Command History
    - ✅History: Previous Command
    - ✅History: Next Command

V1.2 Features Demo

https://github.com/AY2223S1-CS2103T-T13-4/tp/releases/tag/v1.2

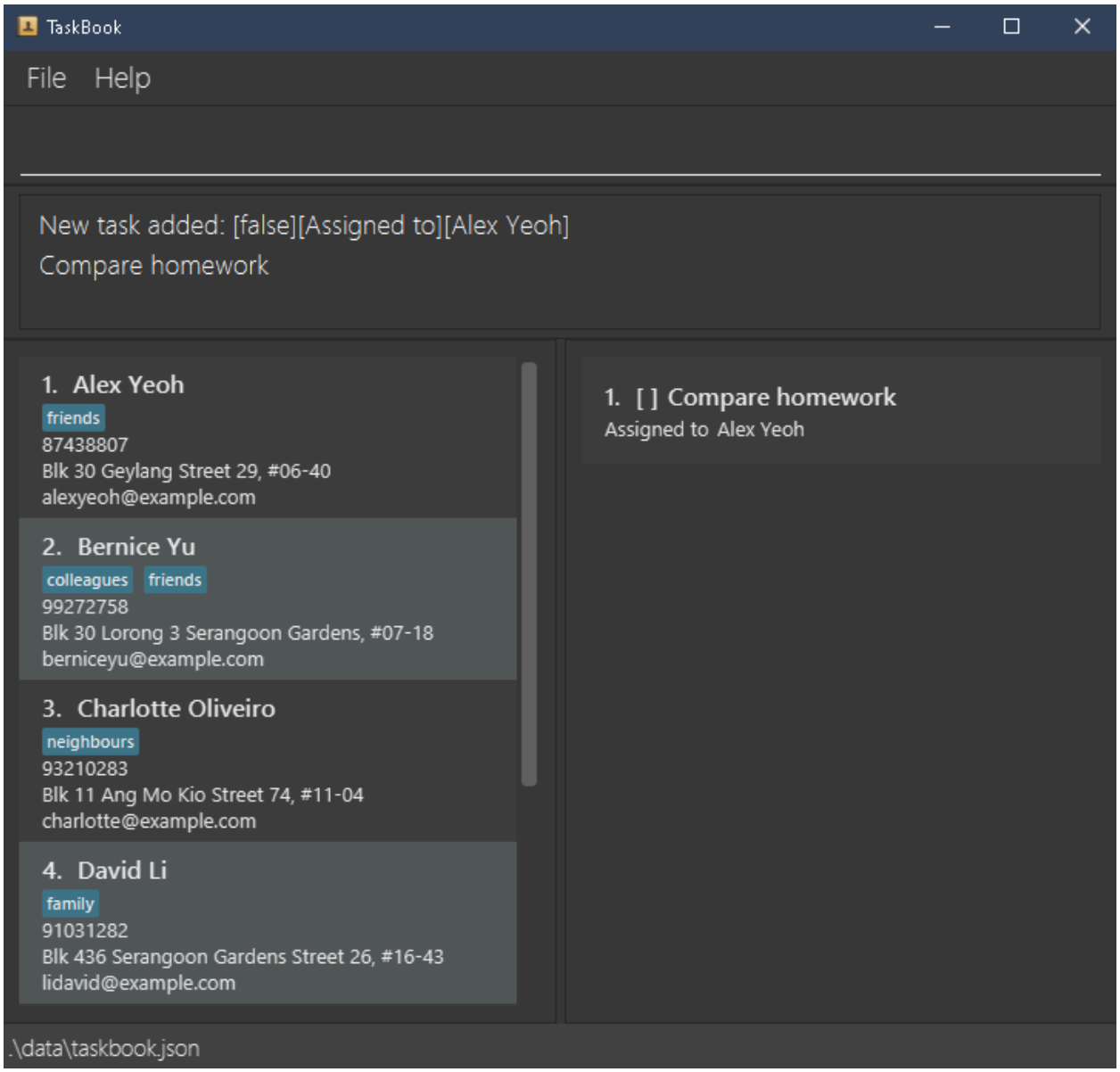
Initial State



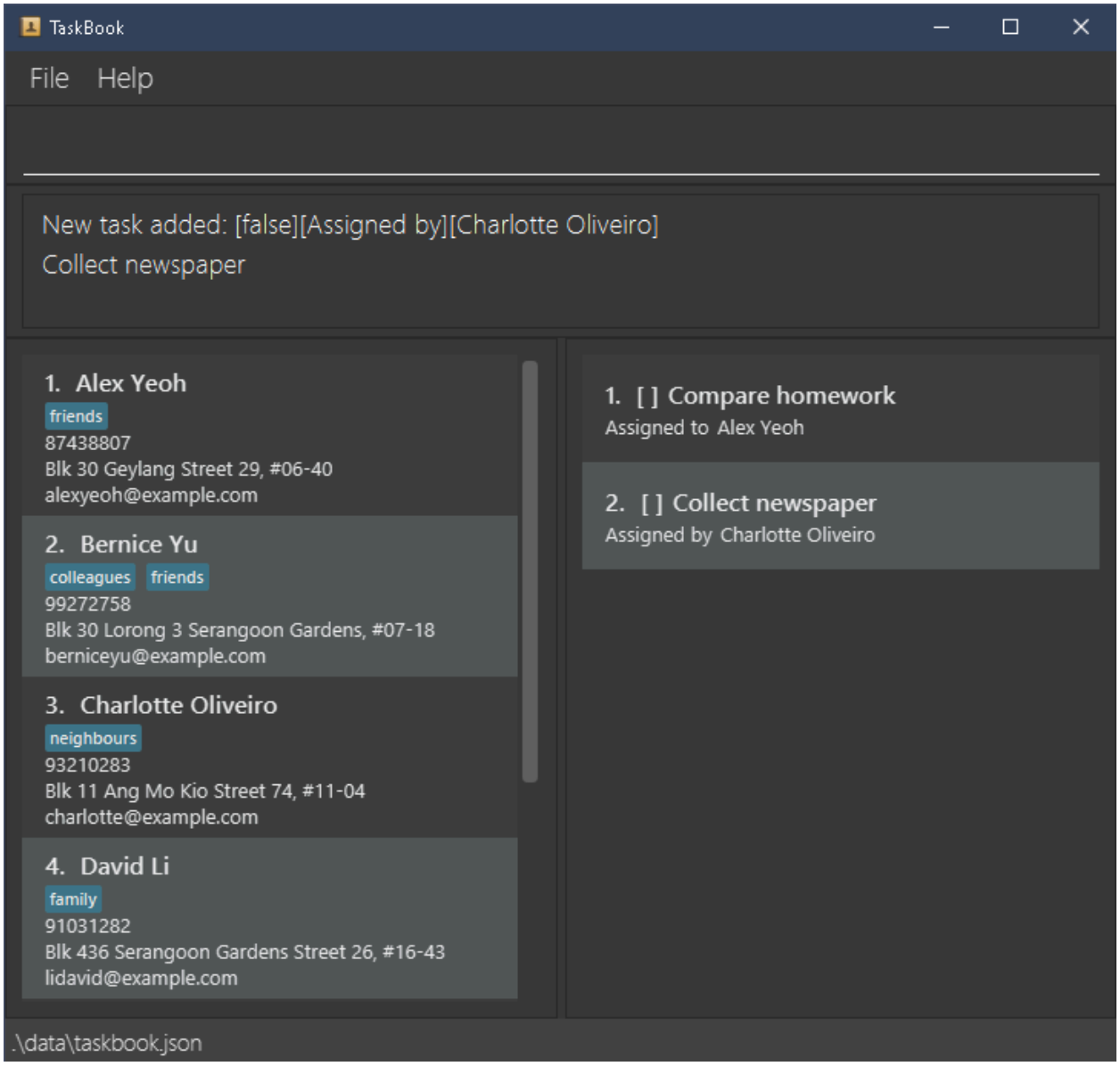
task add

We will first add 7 tasks to demonstrate the scrolling of the window.

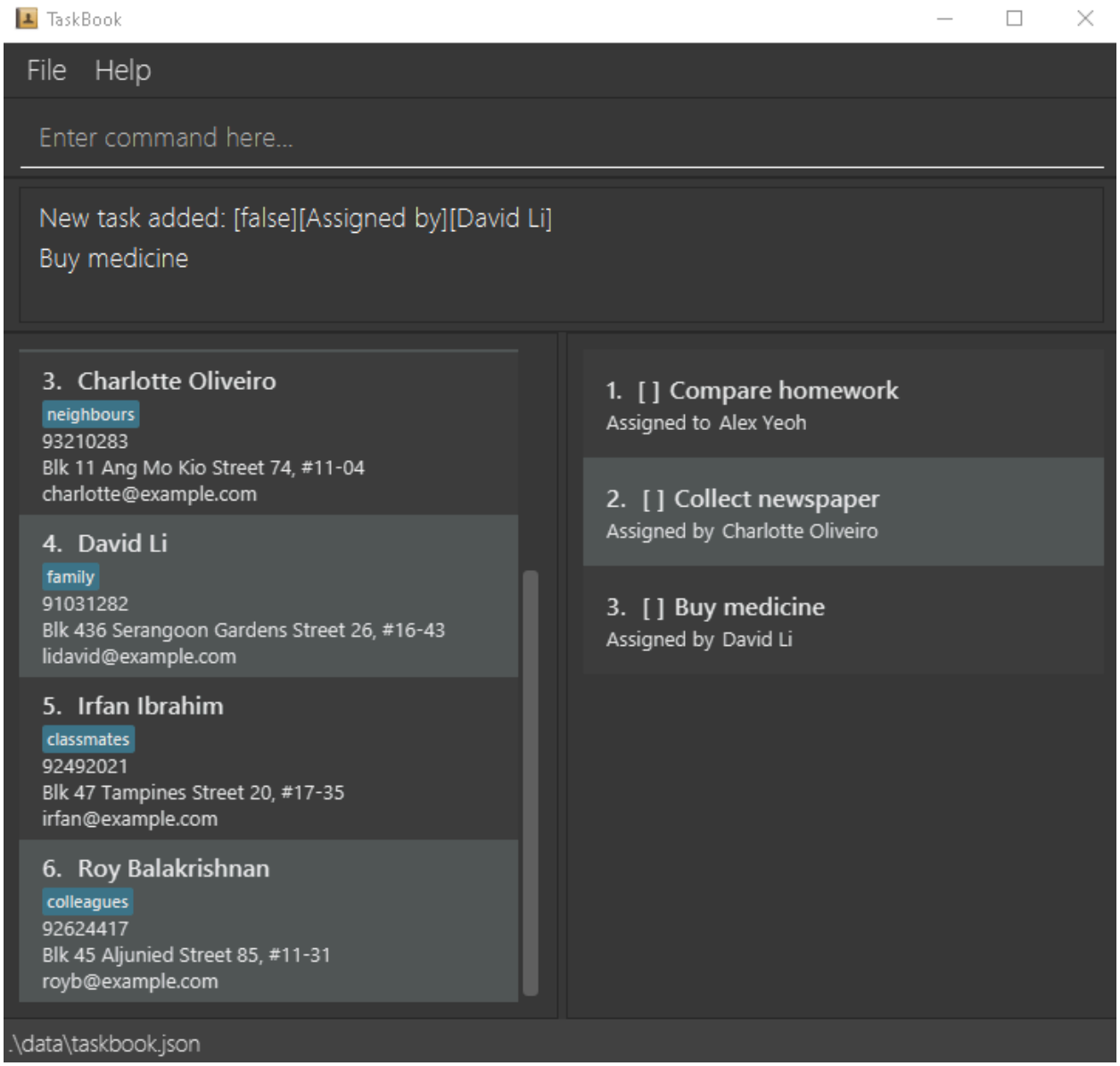
`task add o/Alex Yeoh d/Compare homework`



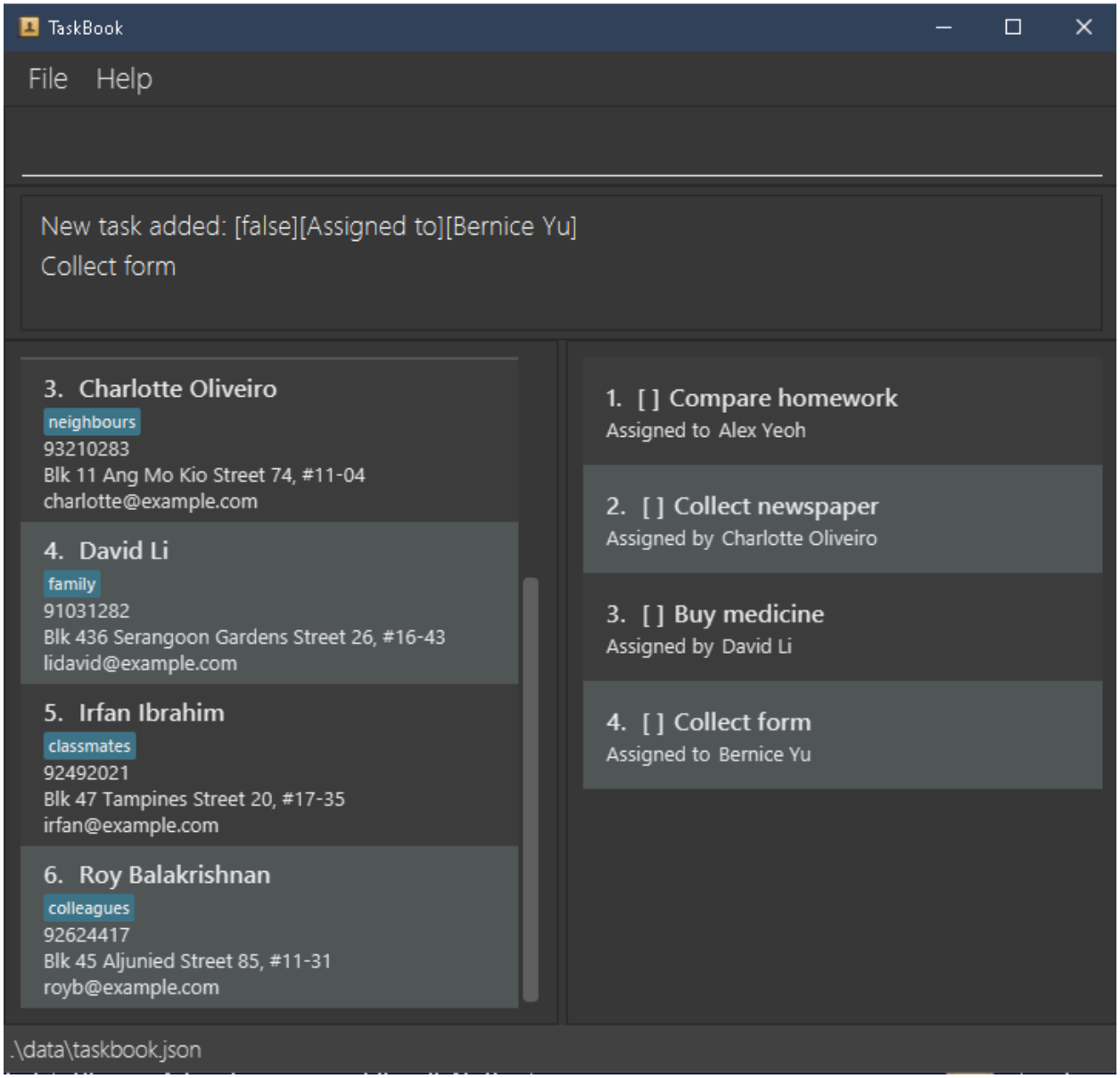
task add m/Charlotte Oliveiro d/Collect newspaper



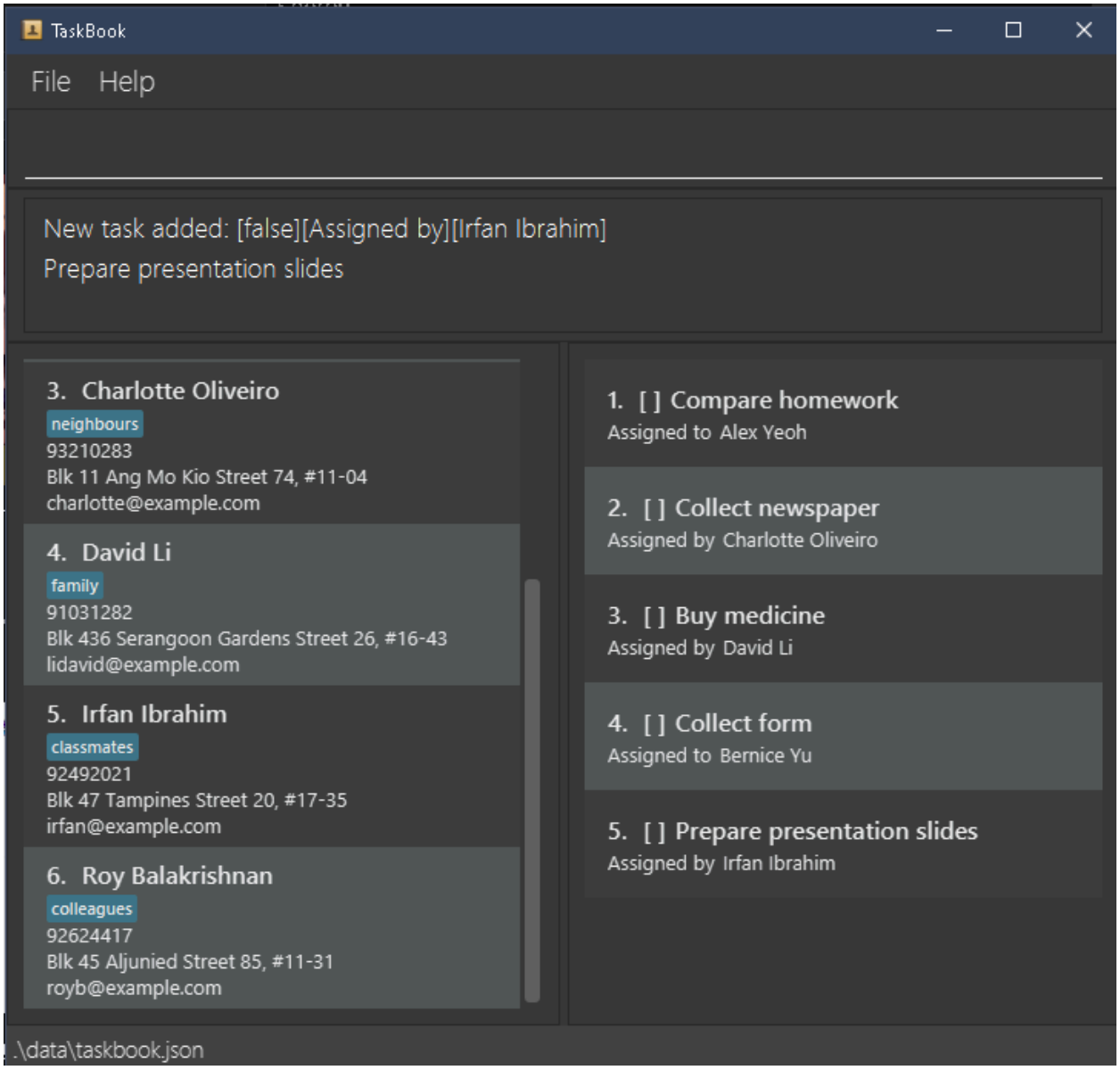
task add m/David Li d/Buy medicine



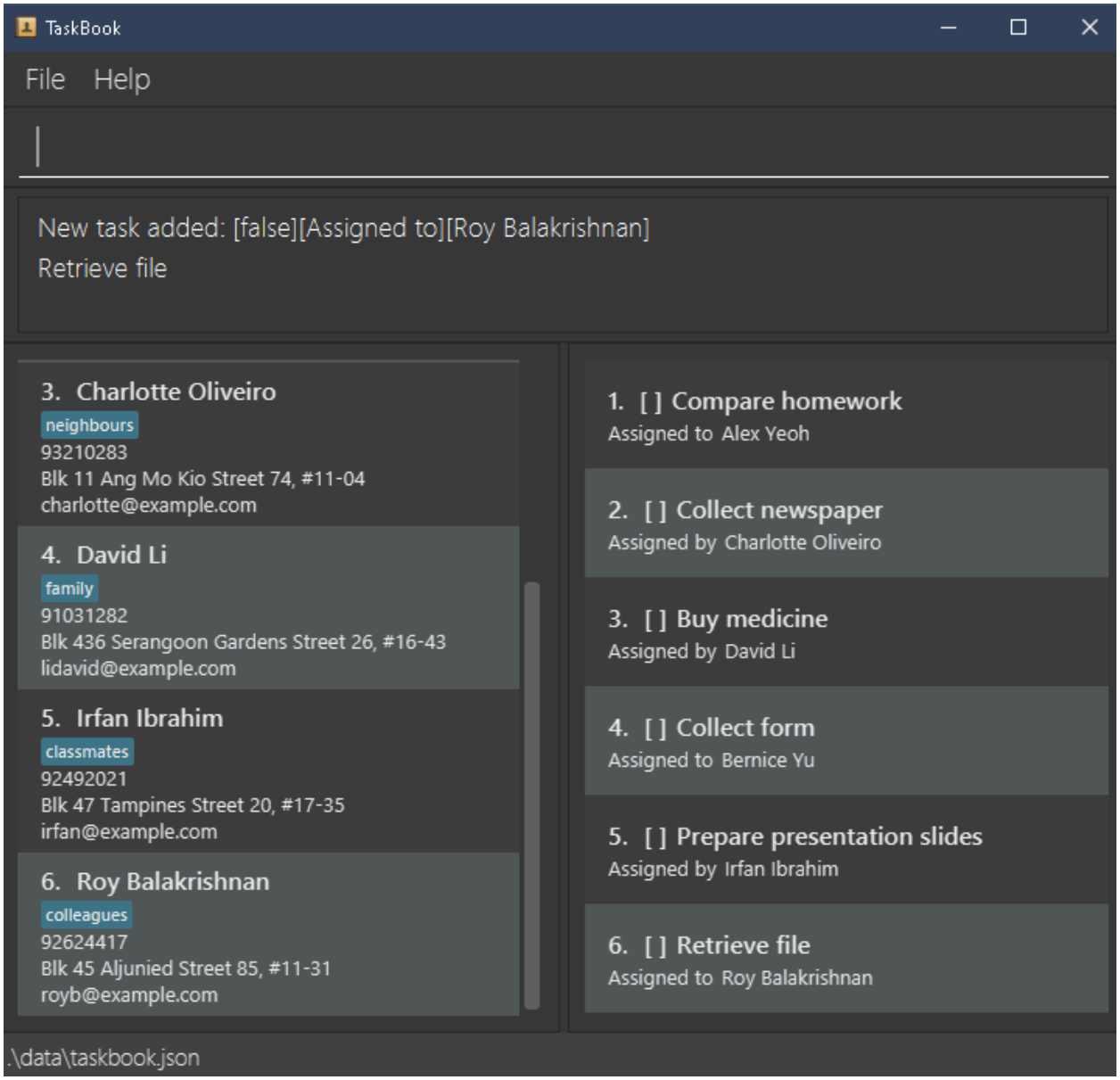
task add o/Bernice Yu d/Collect form



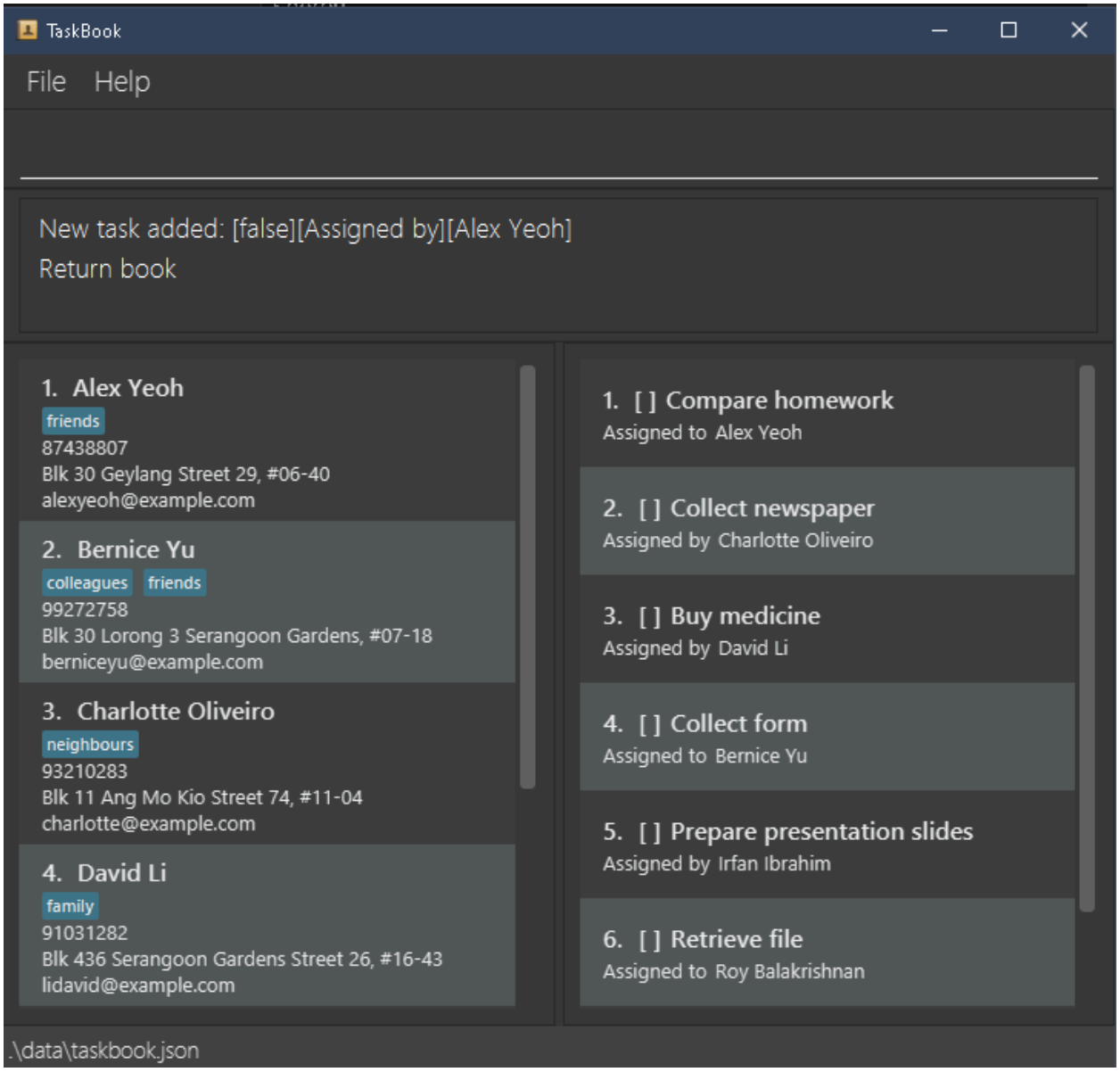
task add m/Irfan Ibrahim d/Prepare presentation slides



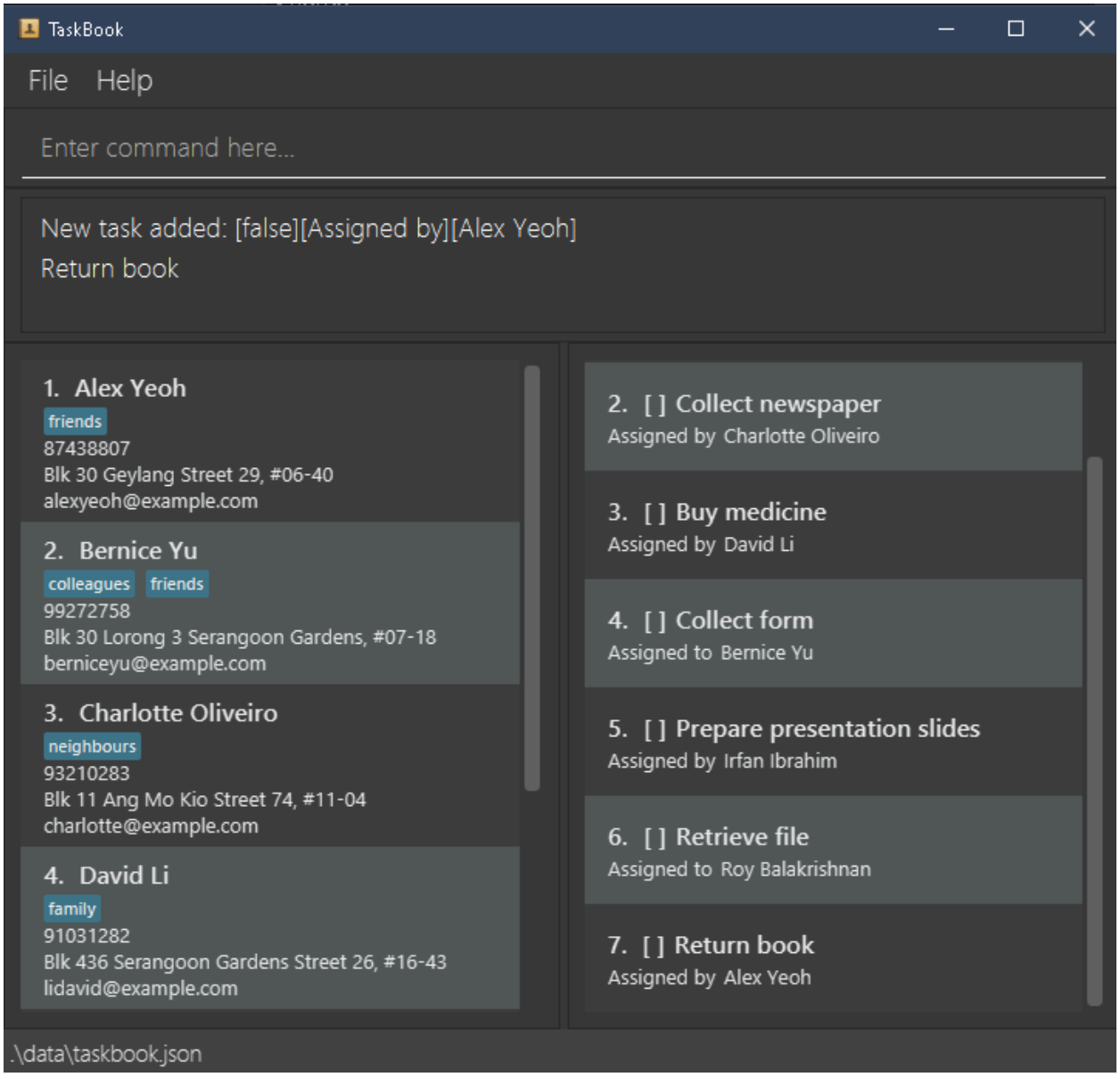
task add o/Roy Balakrishnan d/Retrieve file



task add m/Alex Yeoh d/Return book



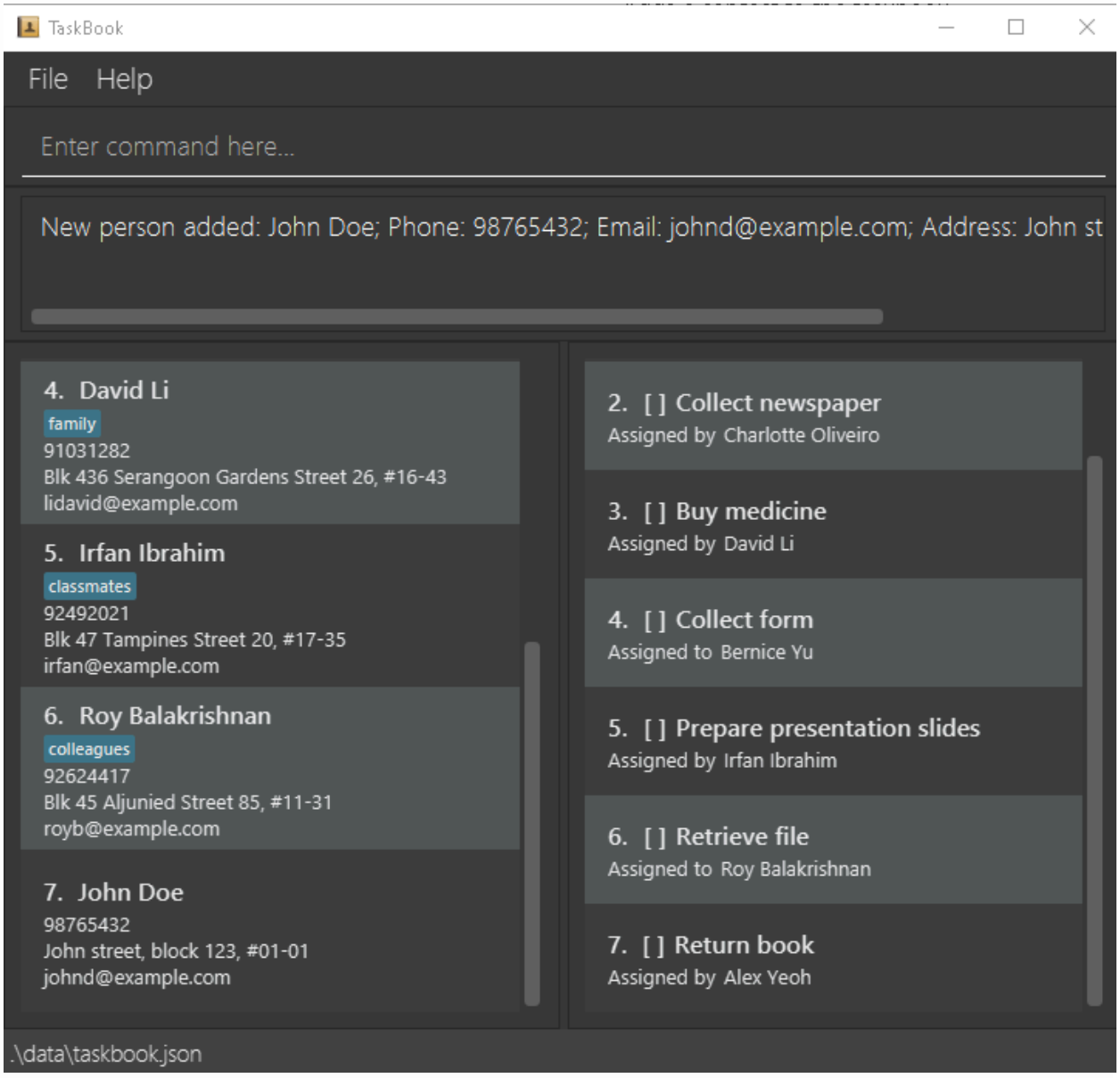
Notice how you can scroll down if the number of tasks exceeds the window.



contact add

We now add a contact

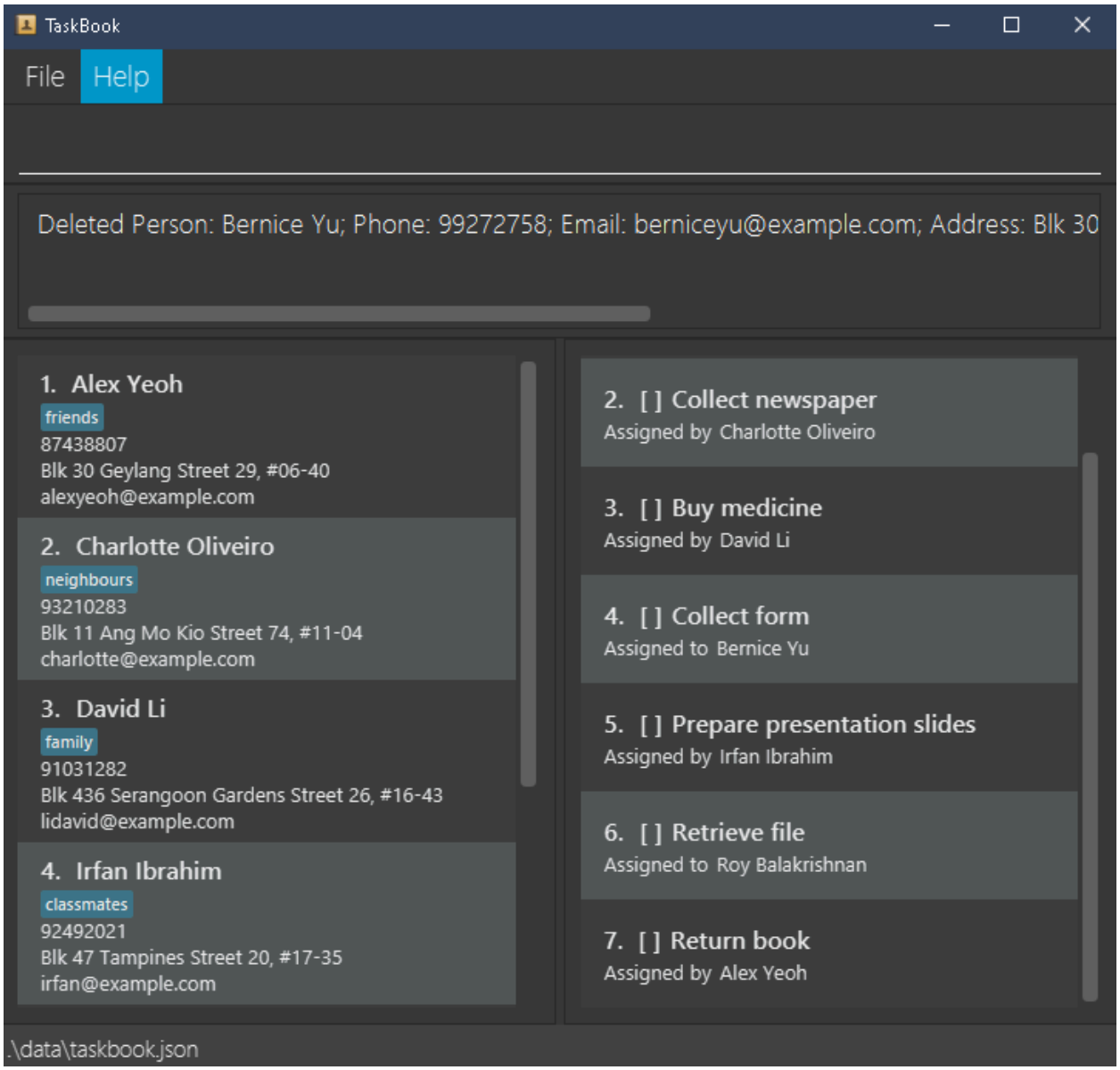
`contact add n/John Doe p/98765432 e/johnd@example.com a/John street, block 123, #01-01`



contact delete

We now delete a contact.

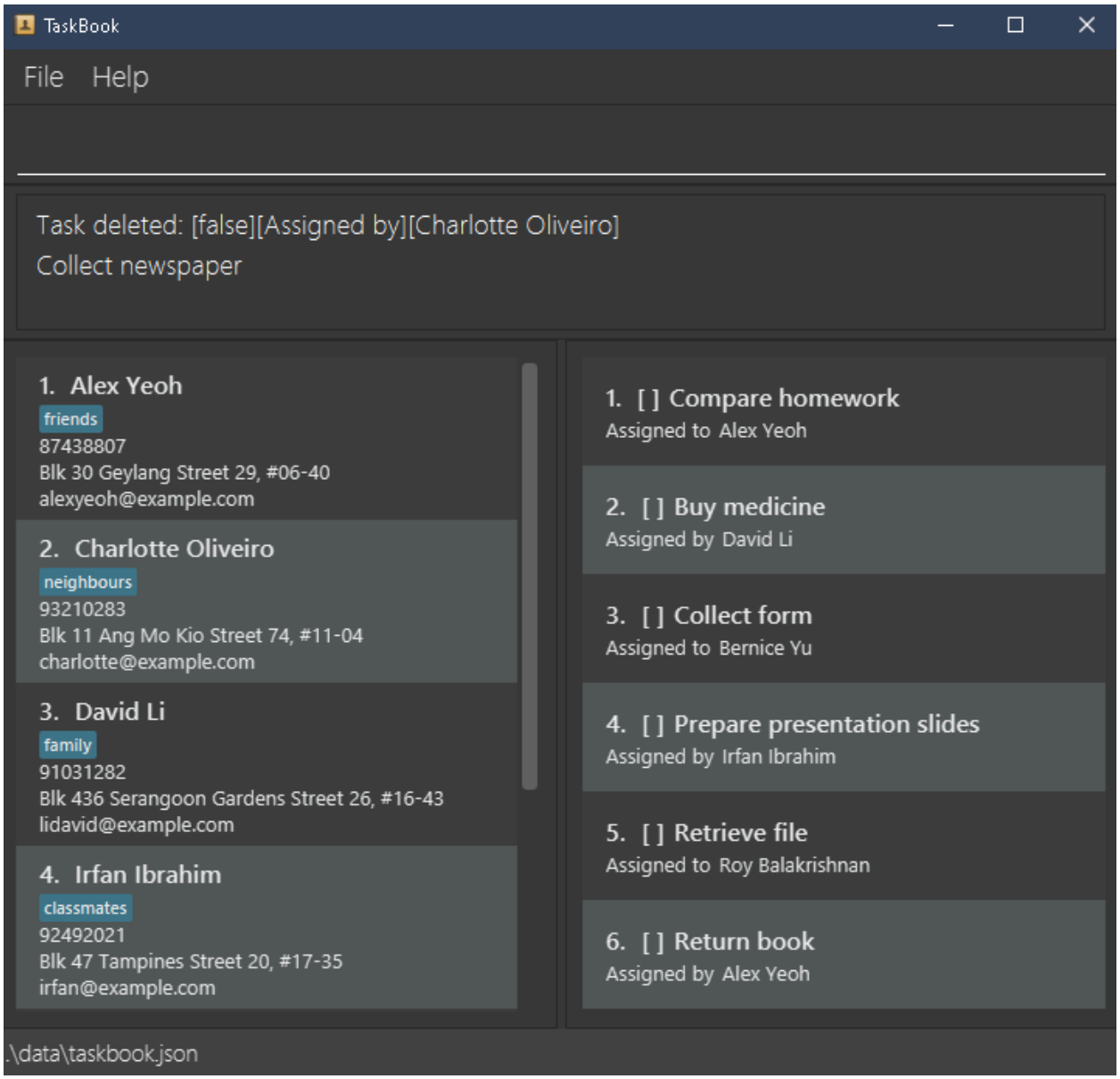
contact delete i/2



task delete

We now delete a task

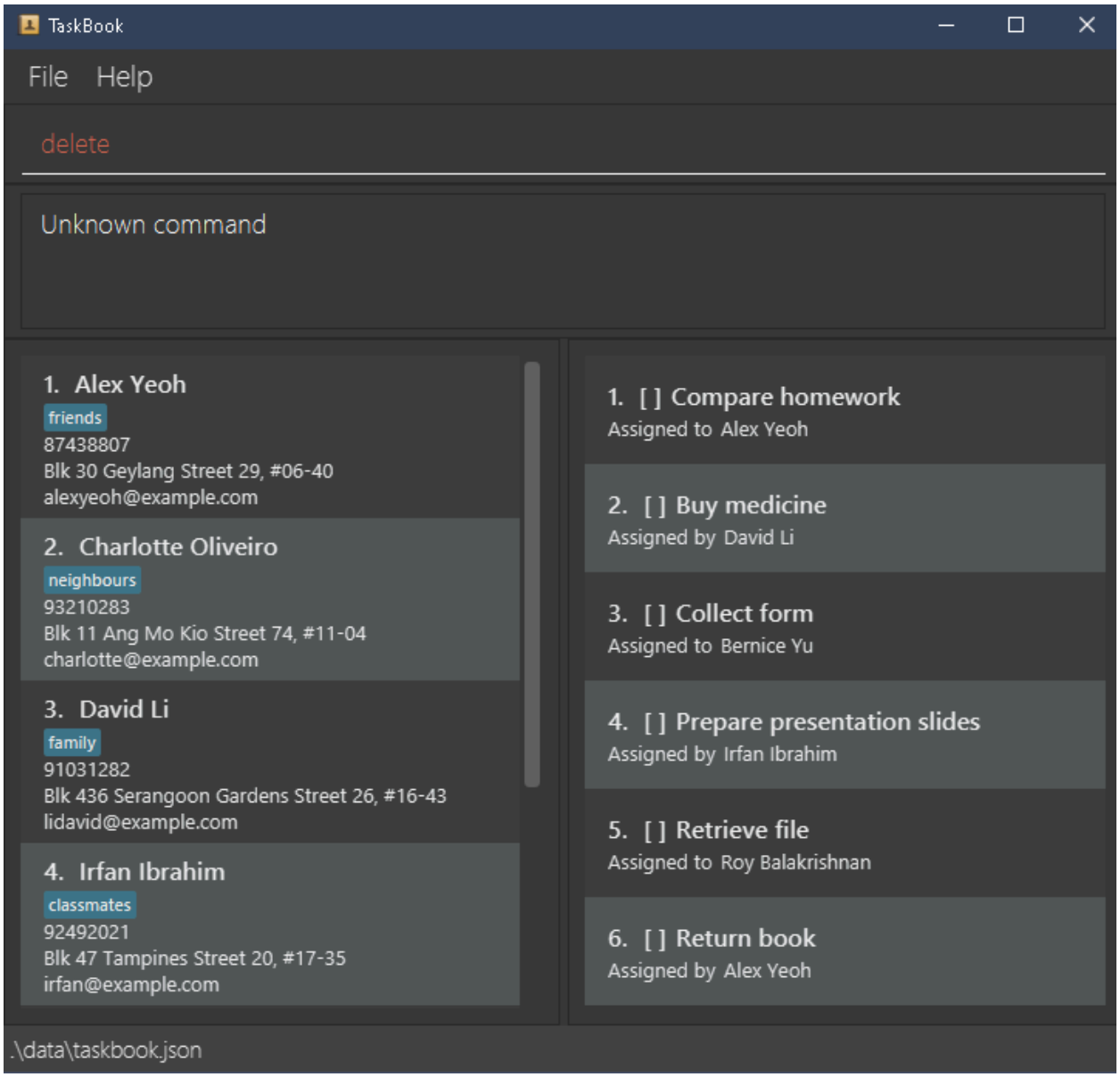
`task delete i/2`



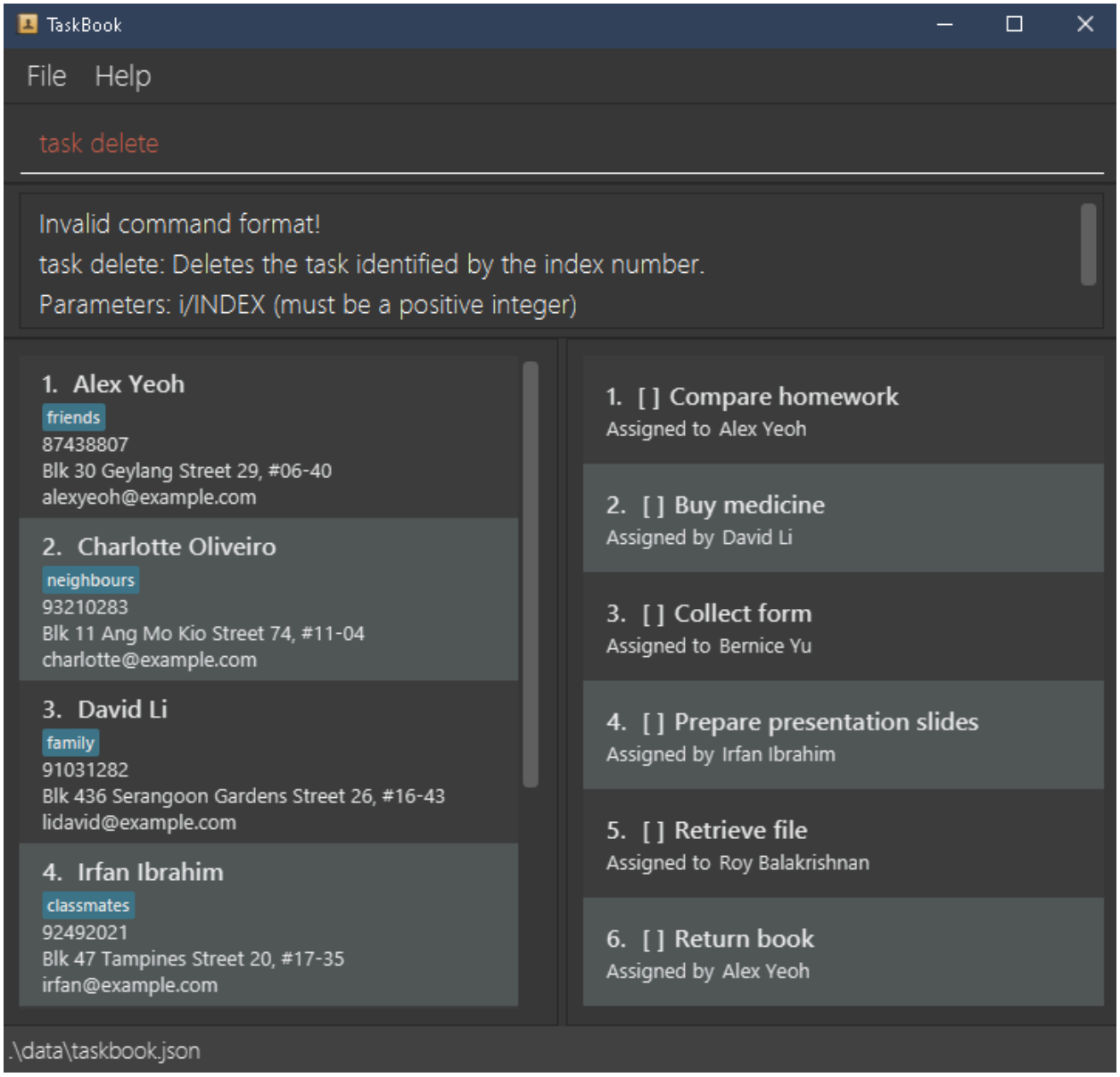
Handled Exceptions

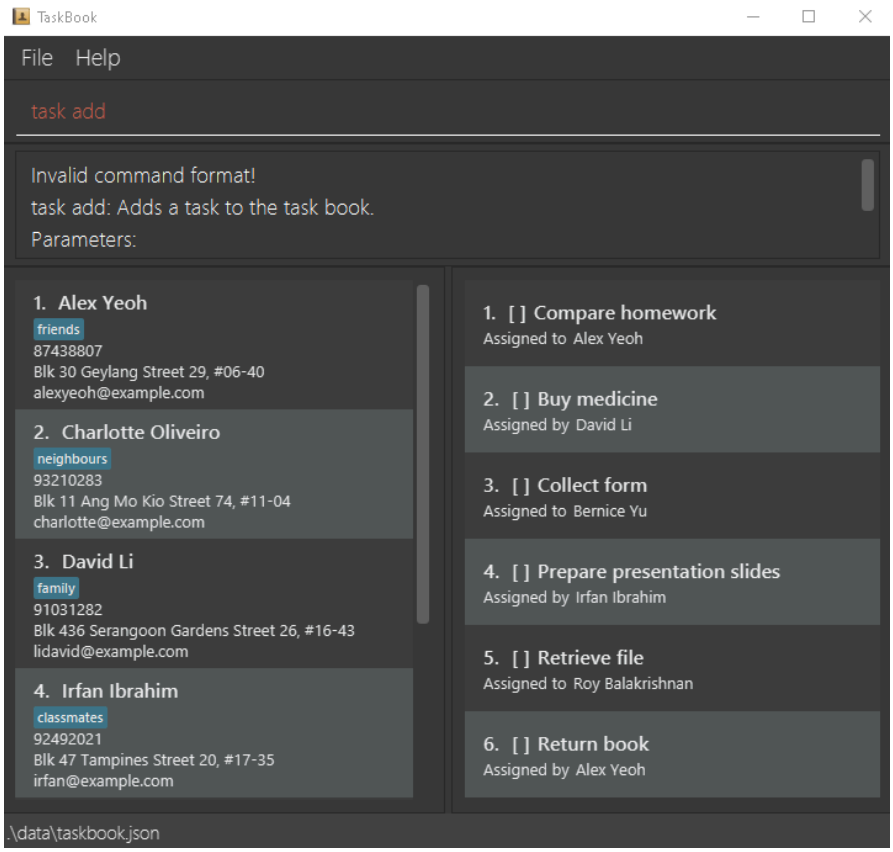
Below are the exceptions for command inputs that we have checked for.

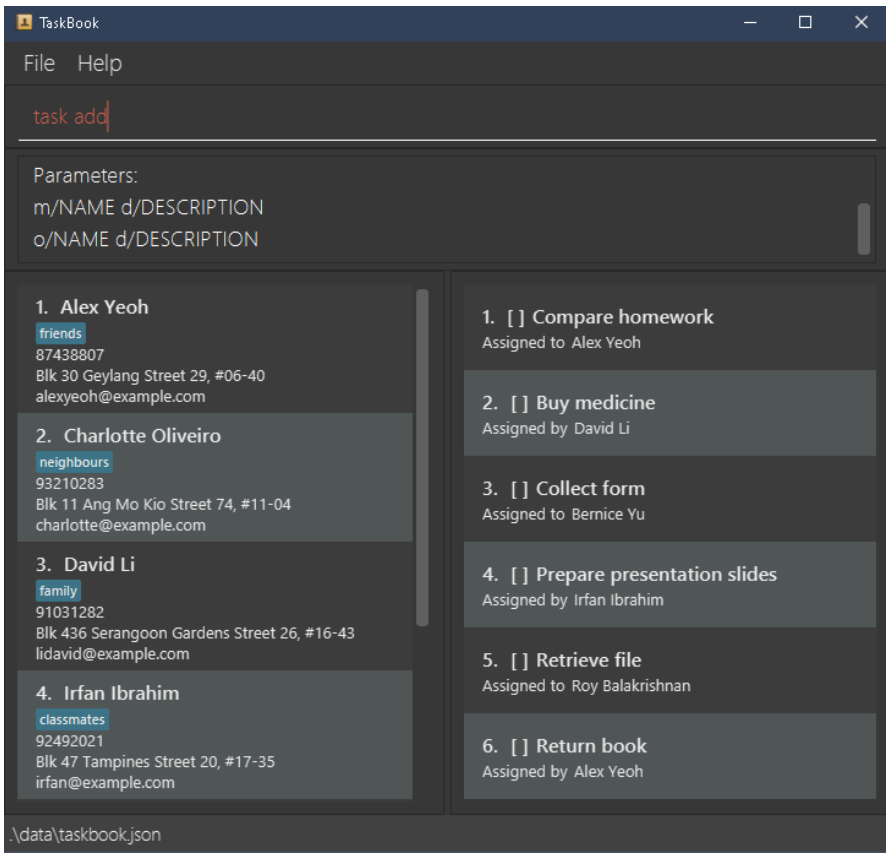
If command input not recognised



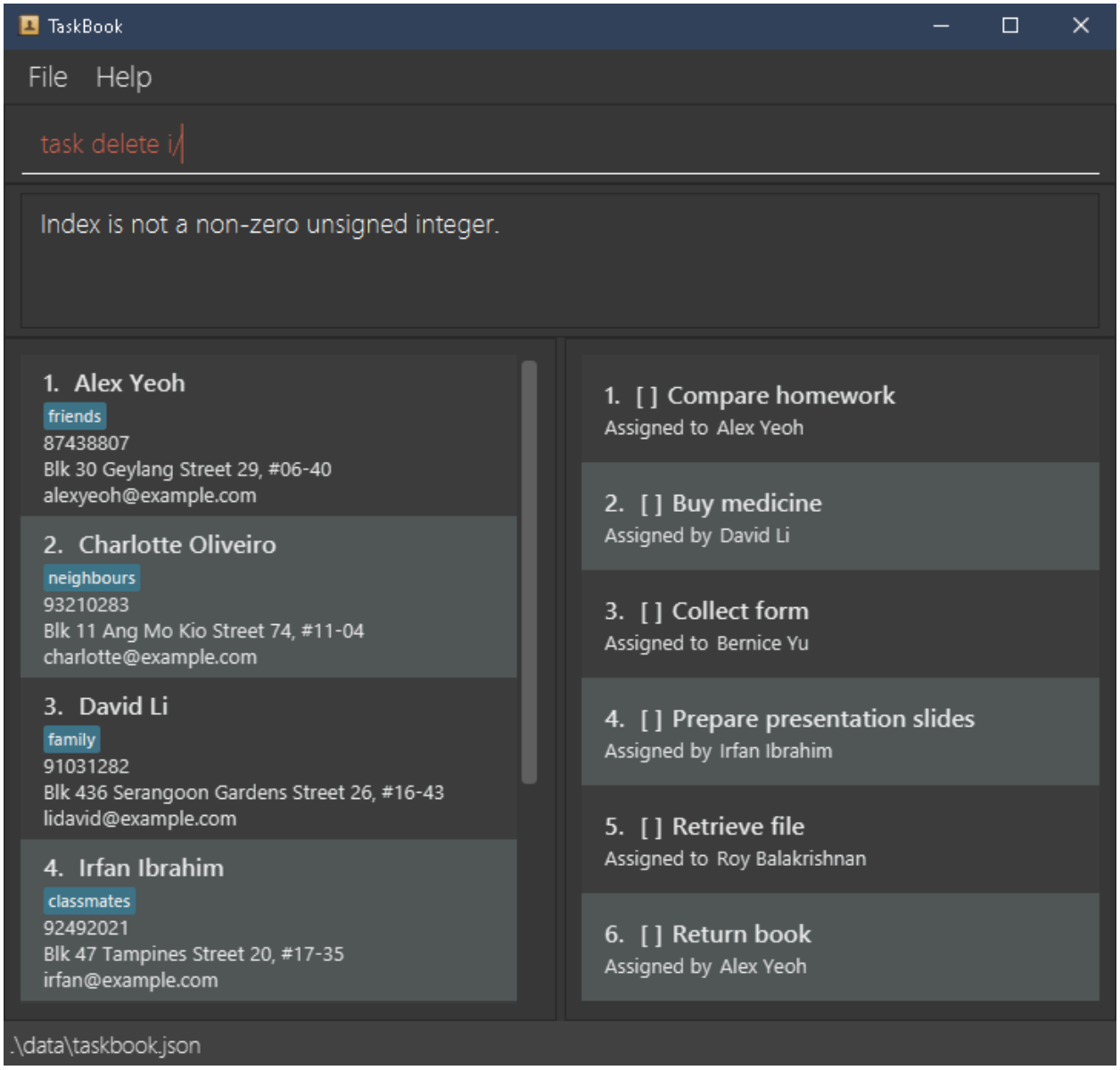
If command given is understood but the input data is of an invalid format



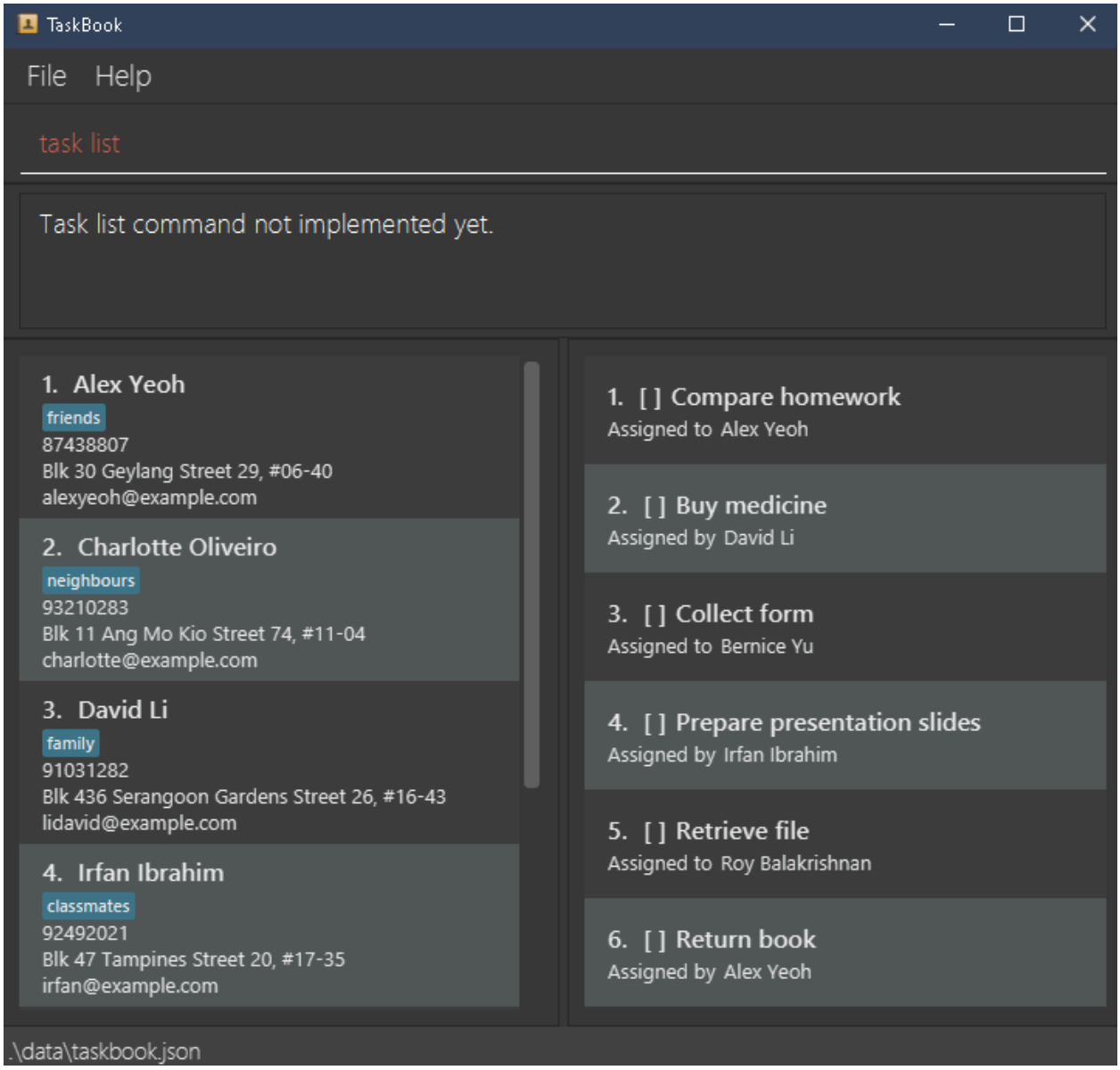

TaskBook
File Help
task add
Invalid command format!
task add: Adds a task to the task book.
Parameters:
1. Alex Yeoh
friends
87438807
Blk 30 Geylang Street 29, #06-40
alexyeoh@example.com
2. Charlotte Oliveiro
neighbours
93210283
Blk 11 Ang Mo Kio Street 74, #11-04
charlotte@example.com
3. David Li
family
91031282
Blk 436 Serangoon Gardens Street 26, #16-43
lidavid@example.com
4. Irfan Ibrahim
classmates
92492021
Blk 47 Tampines Street 20, #17-35
irfan@example.com
1. [ ] Compare homework
Assigned to Alex Yeoh
2. [ ] Buy medicine
Assigned by David Li
3. [ ] Collect form
Assigned to Bernice Yu
4. [ ] Prepare presentation slides
Assigned by Irfan Ibrahim
5. [ ] Retrieve file
Assigned to Roy Balakrishnan
6. [ ] Return book
Assigned by Alex Yeoh
.\data\taskbook.json


TaskBook
File Help
task add
Parameters:
m/NAME d/DESCRIPTION
o/NAME d/DESCRIPTION
1. Alex Yeoh
friends
87438807
Blk 30 Geylang Street 29, #06-40
alexyeoh@example.com
2. Charlotte Oliveiro
neighbours
93210283
Blk 11 Ang Mo Kio Street 74, #11-04
charlotte@example.com
3. David Li
family
91031282
Blk 436 Serangoon Gardens Street 26, #16-43
lidavid@example.com
4. Irfan Ibrahim
classmates
92492021
Blk 47 Tampines Street 20, #17-35
irfan@example.com
1. [ ] Compare homework
Assigned to Alex Yeoh
2. [ ] Buy medicine
Assigned by David Li
3. [ ] Collect form
Assigned to Bernice Yu
4. [ ] Prepare presentation slides
Assigned by Irfan Ibrahim
5. [ ] Retrieve file
Assigned to Roy Balakrishnan
6. [ ] Return book
Assigned by Alex Yeoh
.\data\taskbook.json

If the command and format is valid, but the actual inputs are invalid



task list - not implemented yet



bye

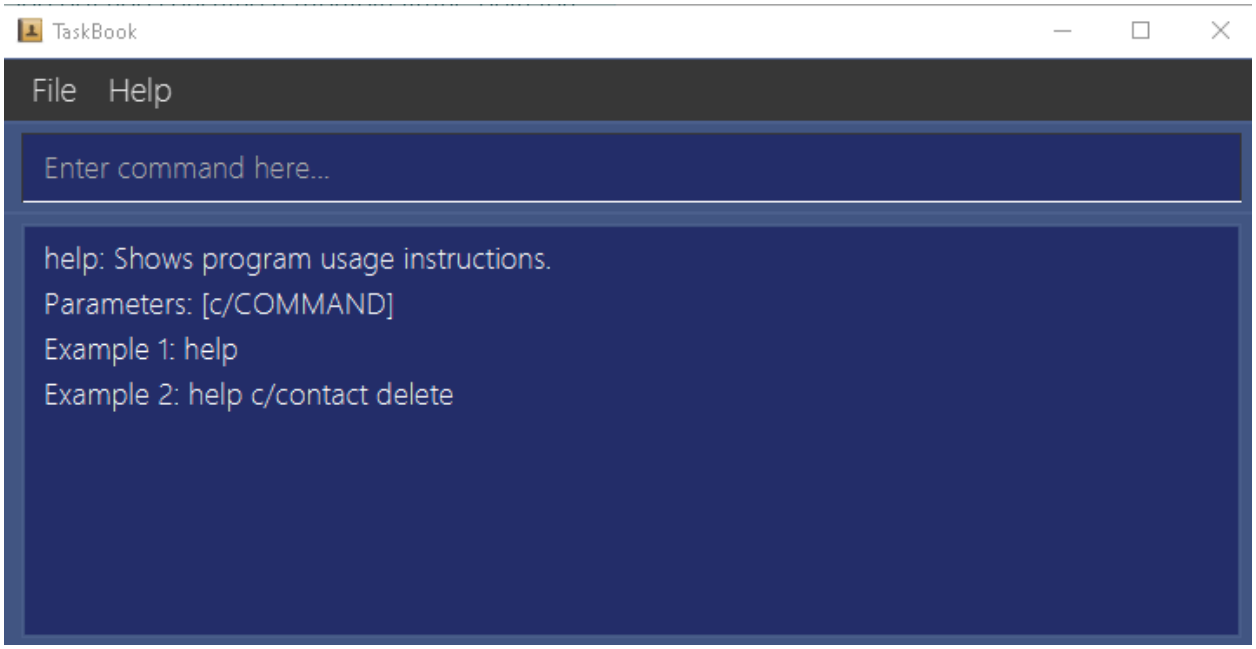
bye stops the application and closes the window immediately.

V1.3 Features Demo

Help command

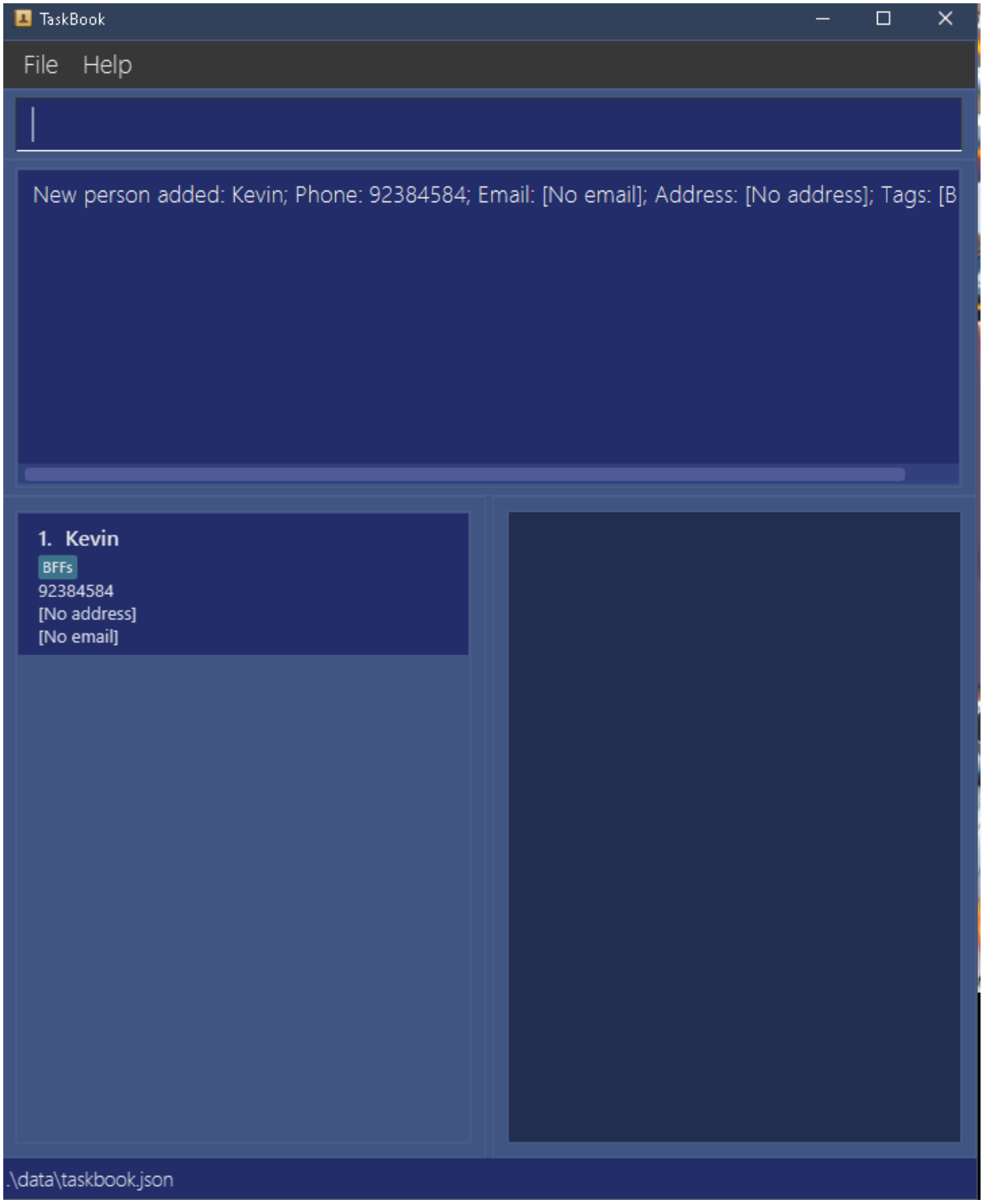
`help`

`help c/help`



Adding contacts

We first add a few contacts.
`contact add n/Kevin p/92384584 #/BFFs`



After executing:
contact add p/98786942 n/Jin Wei e/jinjinweiwei@gmail.com #/Friend
contact add n/Humphrey p/98784269 a/Star Street, Block 123, #01-01 #/Colleague
contact add n/Dexter a/Earth Road, Block 432, #02-02 e/dexterlab@hotmail.com #/Neighbour
contact add n/Bill a/Earth Road, Block 869, #02-02 e/billboard@yahoo.com
contact add n/John #/Scumbag
We will have

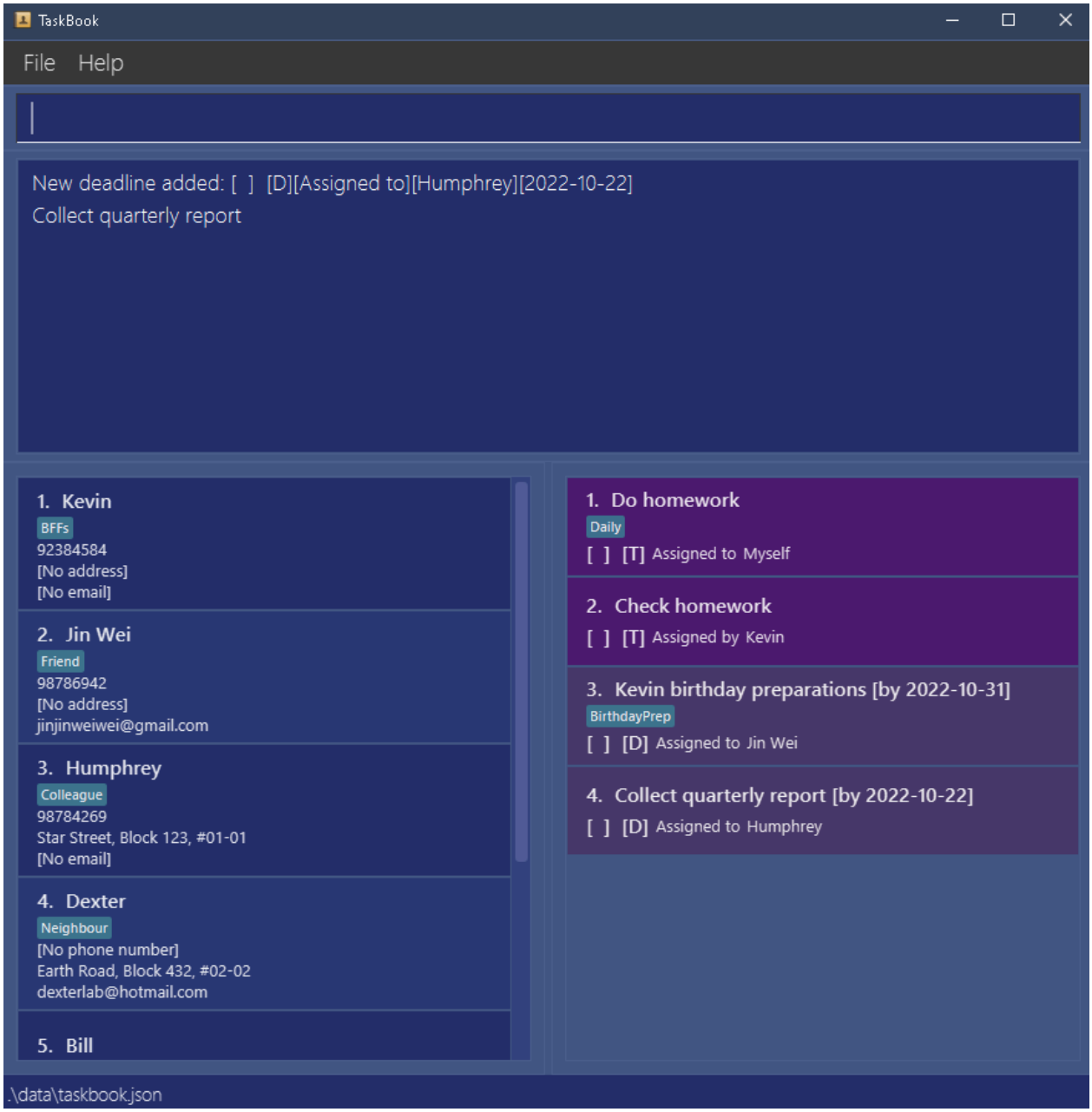
Adding Tasks

There are 3 types of tasks we can add: Todo, Deadline, and Event.
We first add some todos
`task todo d/Do homework #/Daily`
`task todo m/Kevin d/Check homework`

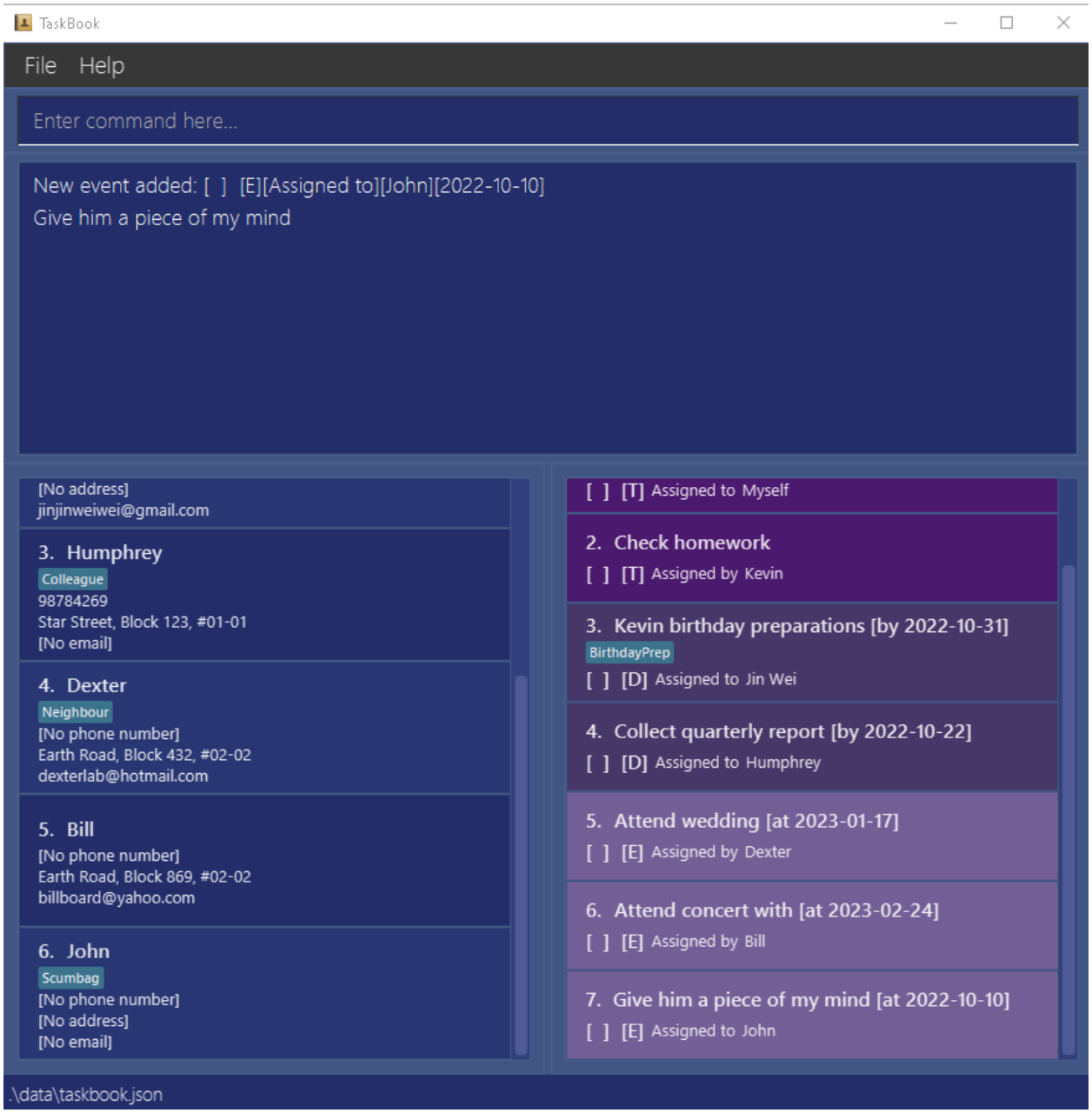
The first is a self assignment because the m/ and o/ flags are absent.

Next, we add 2 deadlines.
task deadline o/Jin Wei d/Kevin birthday preparations t/2022-10-31 #/BirthdayPrep
task deadline o/Humphrey d/Collect quarterly report t/2022-10-22

Notice how the date is cut off in task 3? We can actually increase the window's size!

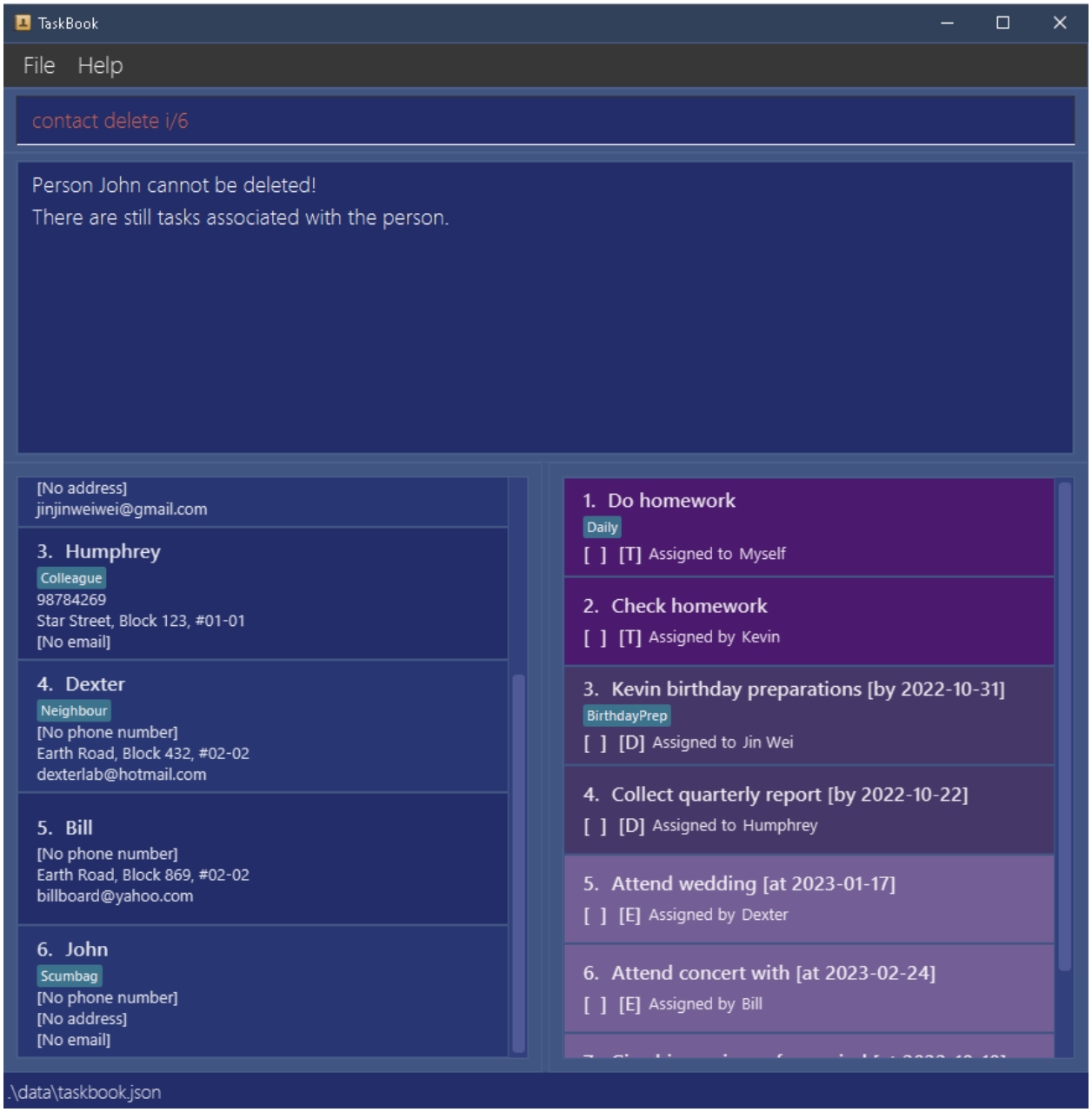


Finally, we add 3 events.
task event m/Dexter d/Attend wedding t/2023-01-17
task event m/Bill d/Attend concert with t/2023-02-24
task event o/John d/Give him a piece of my mind t/2022-10-10

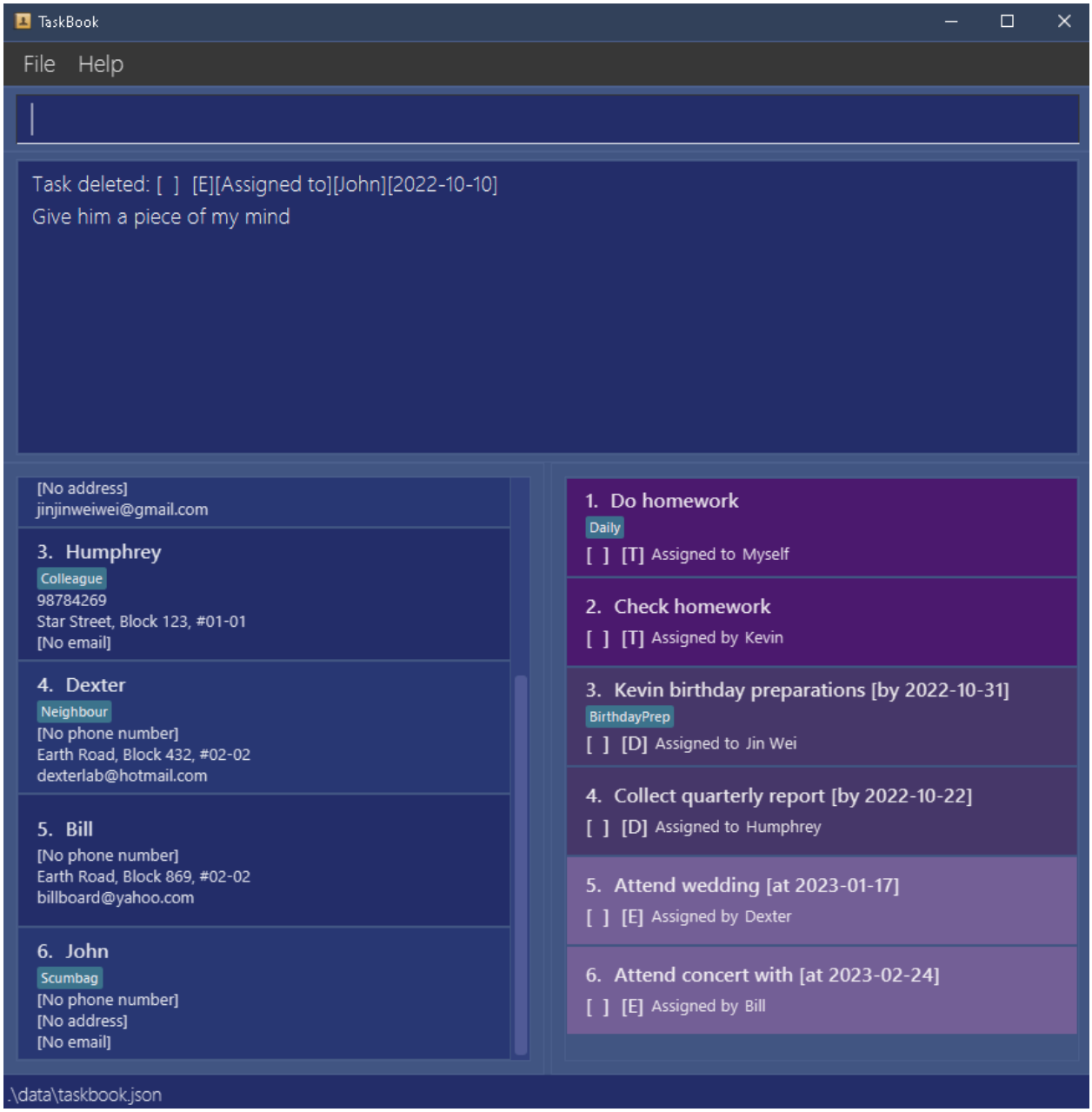


Deleting Tasks and Contacts

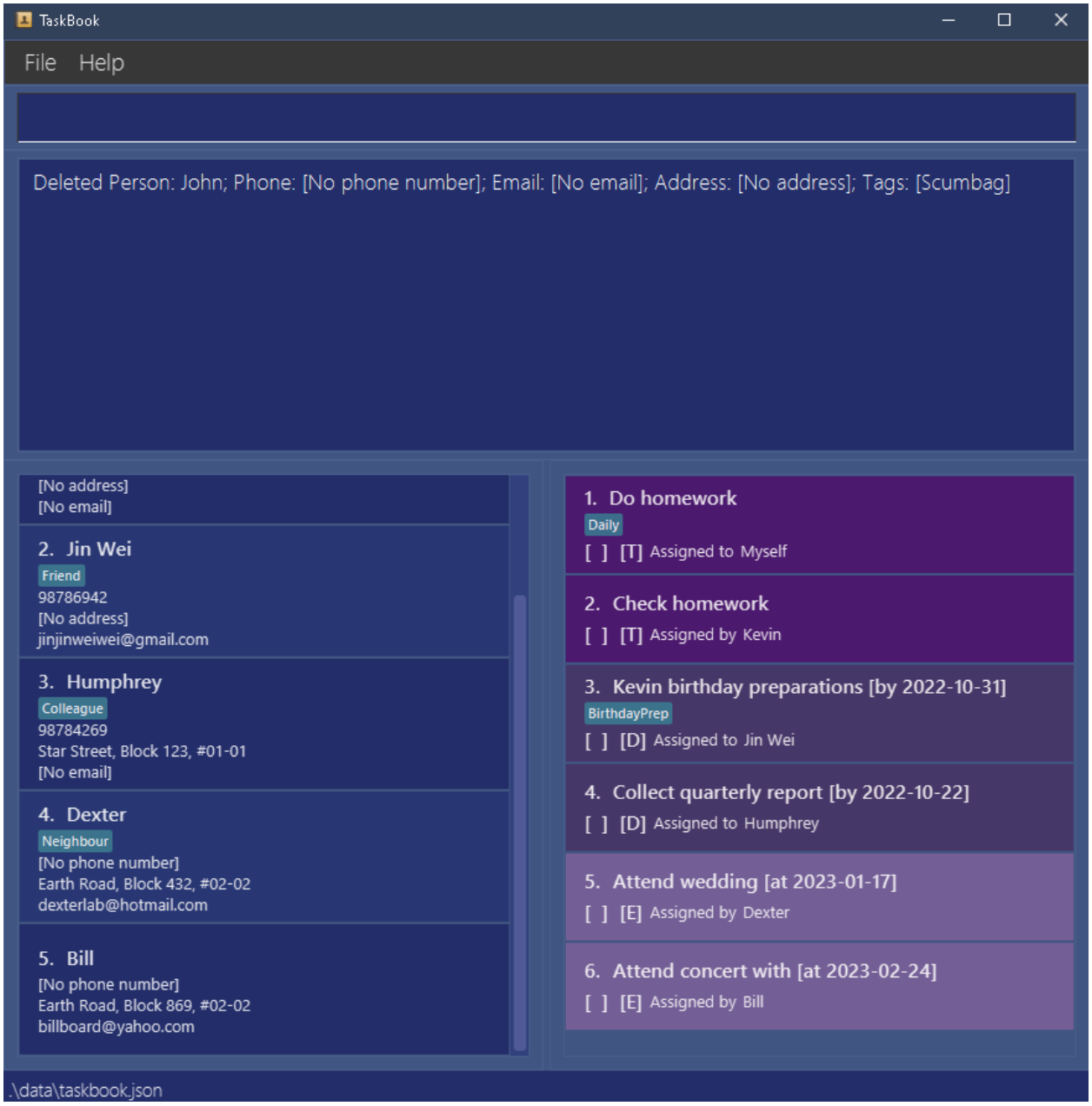
contact delete i/6 will fail, because task 7 is connected to contact 6. We thus need to delete task 7 before deleting contact 6.



Thus, we do task delete i/7 first

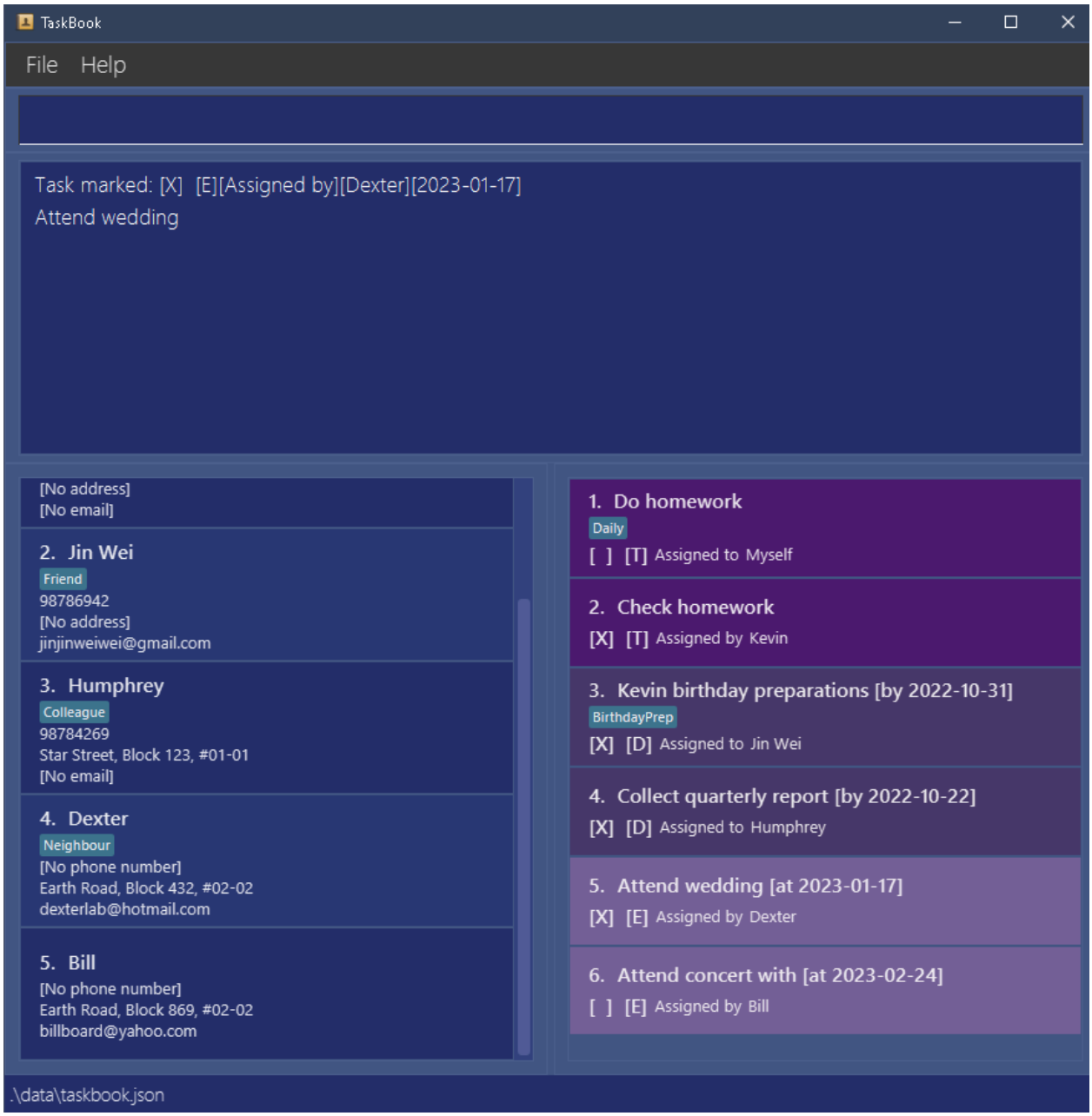


Then we can do contact delete i/6



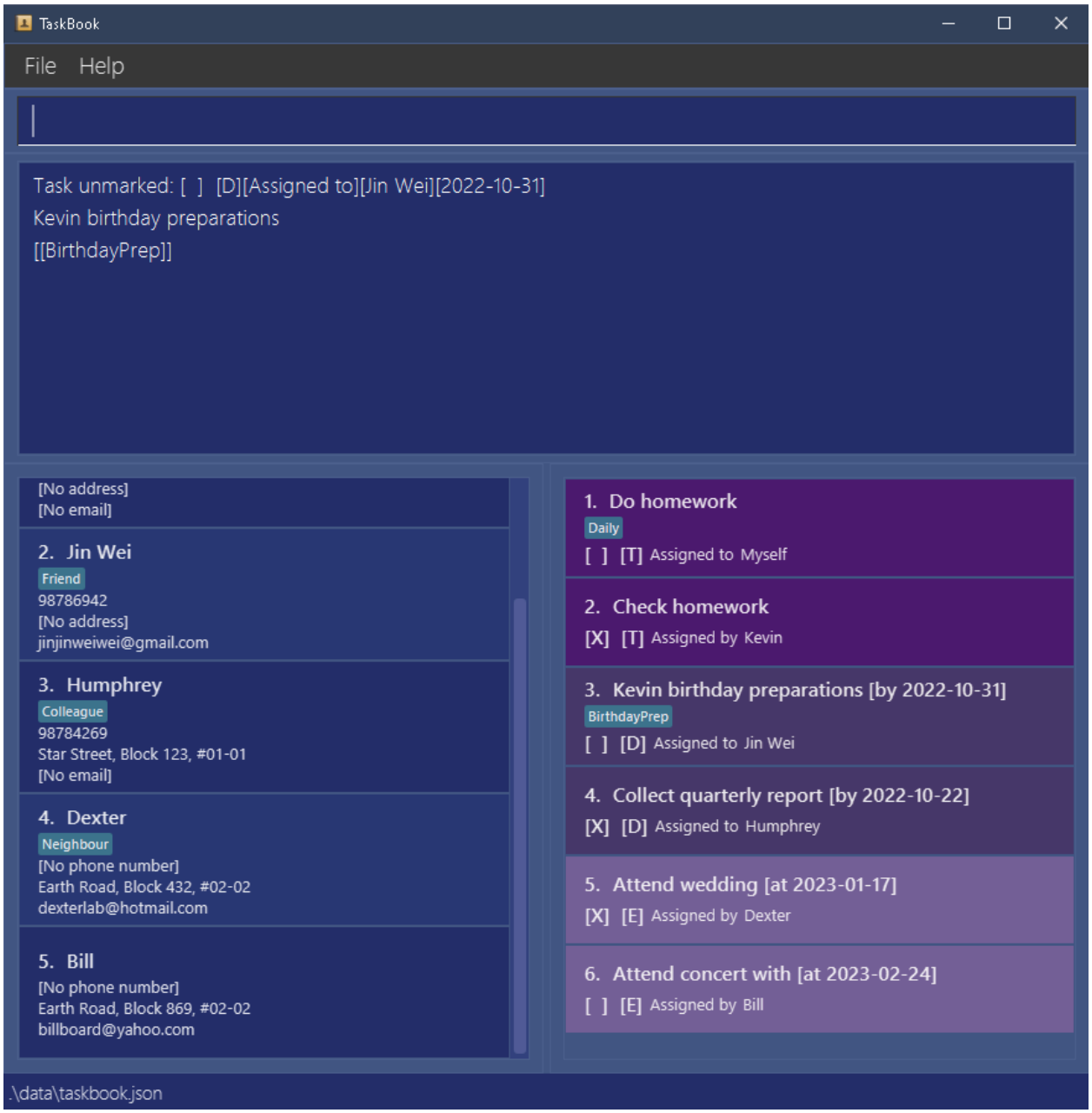
Marking Tasks

task mark i/2
task mark i/3
task mark i/4
task mark i/5



Unmarking Tasks

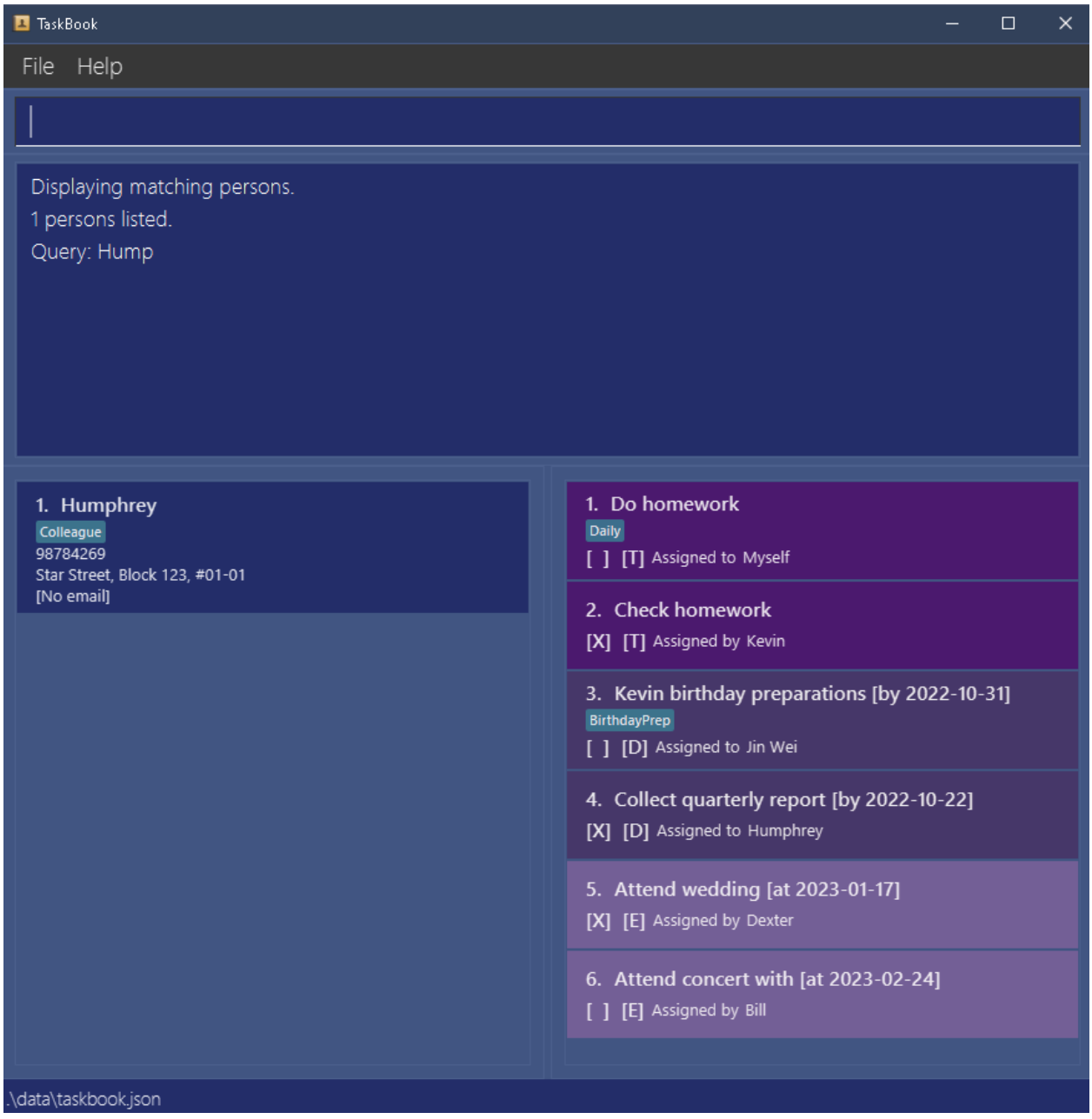
task unmark i/3



Contact Organization

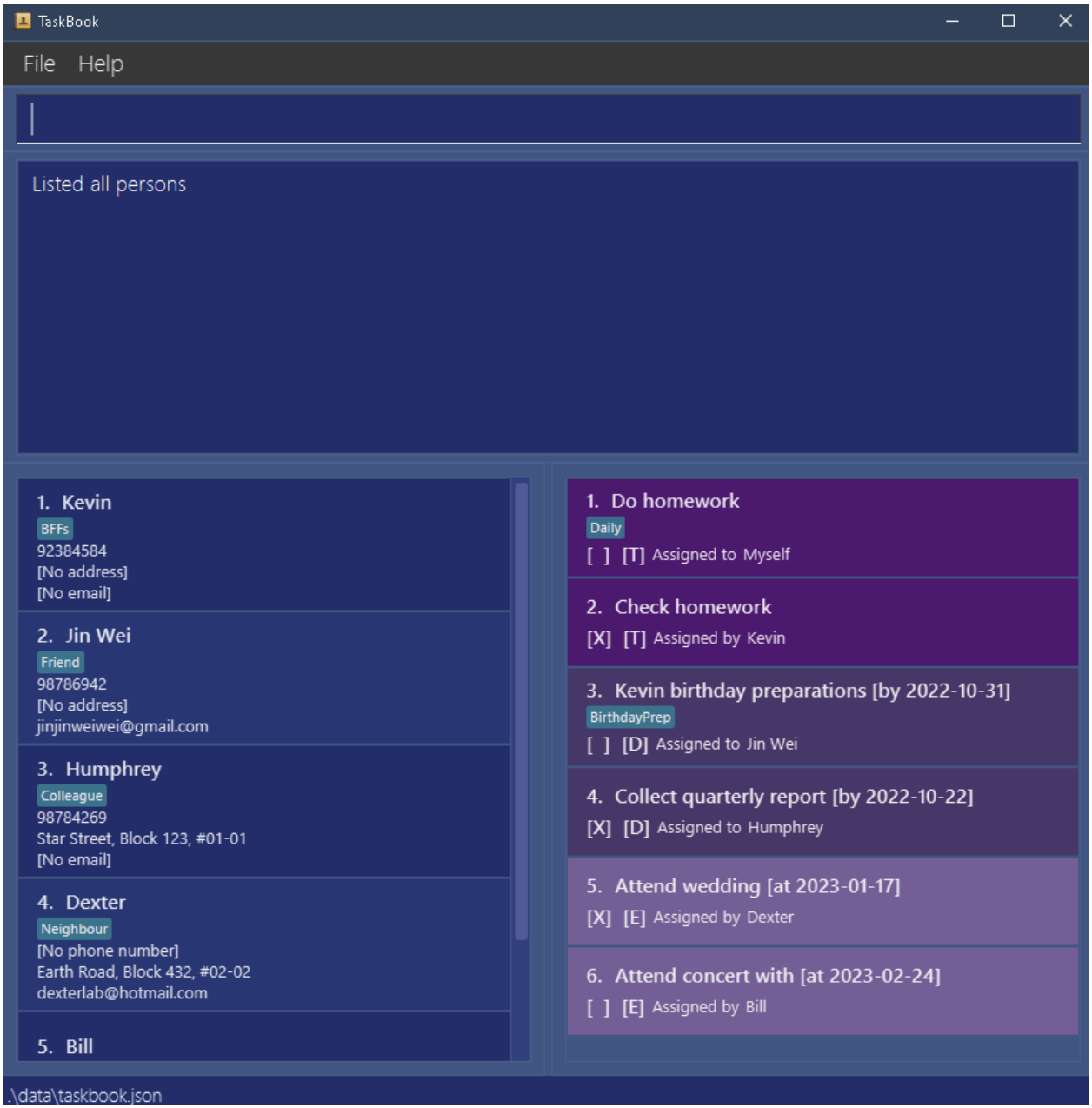
Finding Contacts

contact find q/Hump



Resetting Contact Find Filters

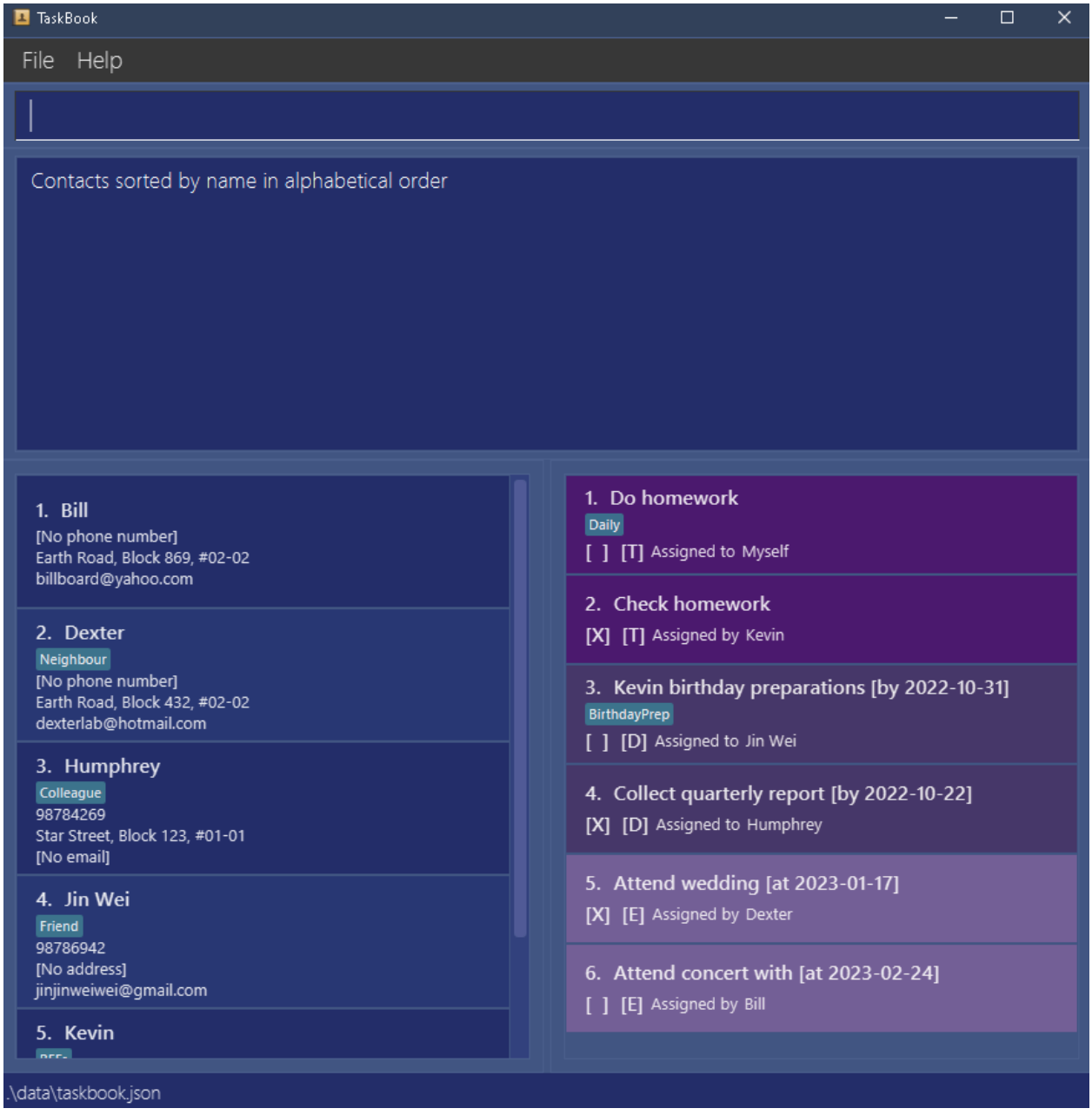
contact list



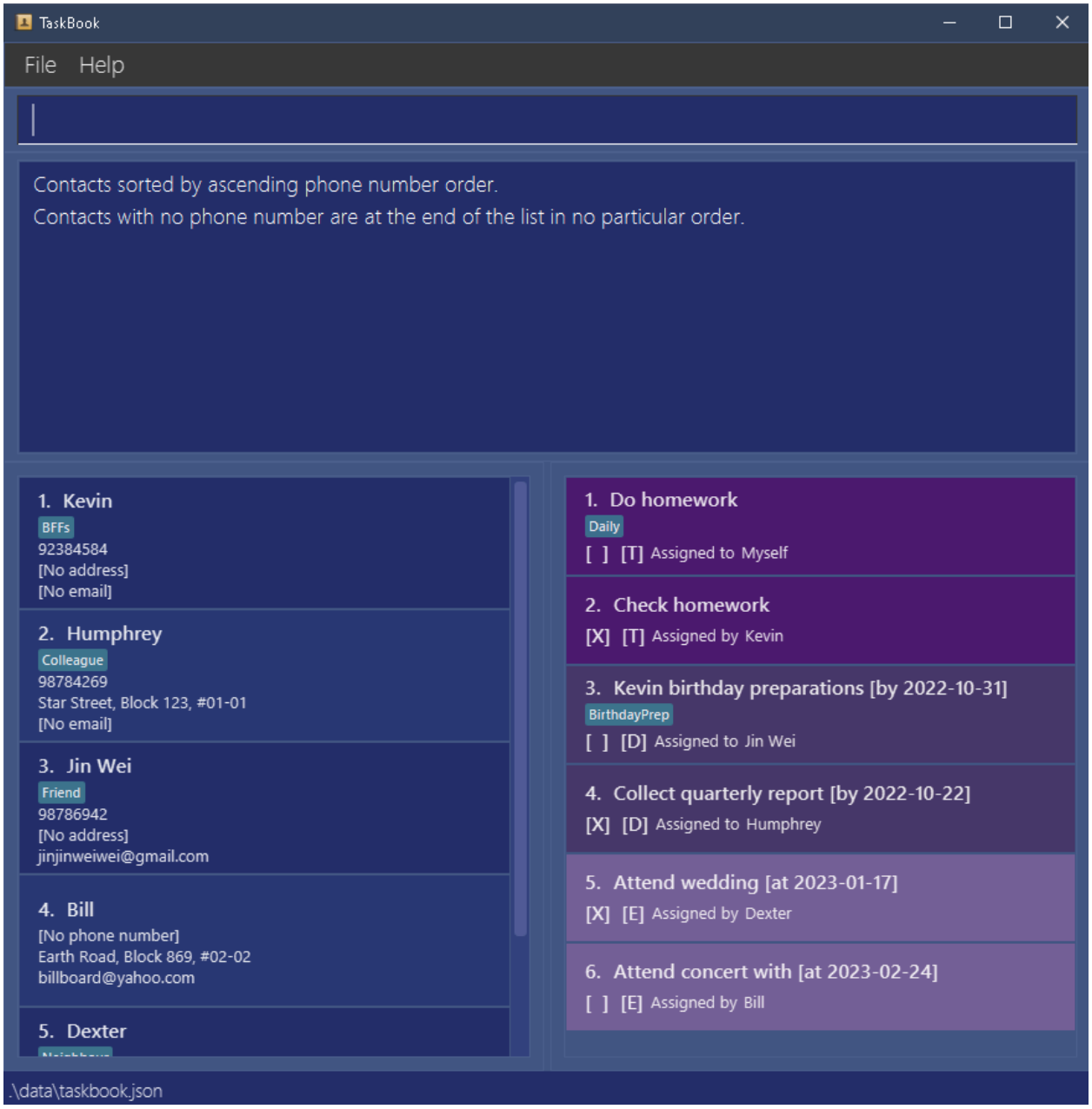
Sorting Contacts

Note: Operations on contact indices, i.e contact delete, work on the index of the contact that is being displayed on the GUI, not the actual original list. This is the more intuitive implementation.

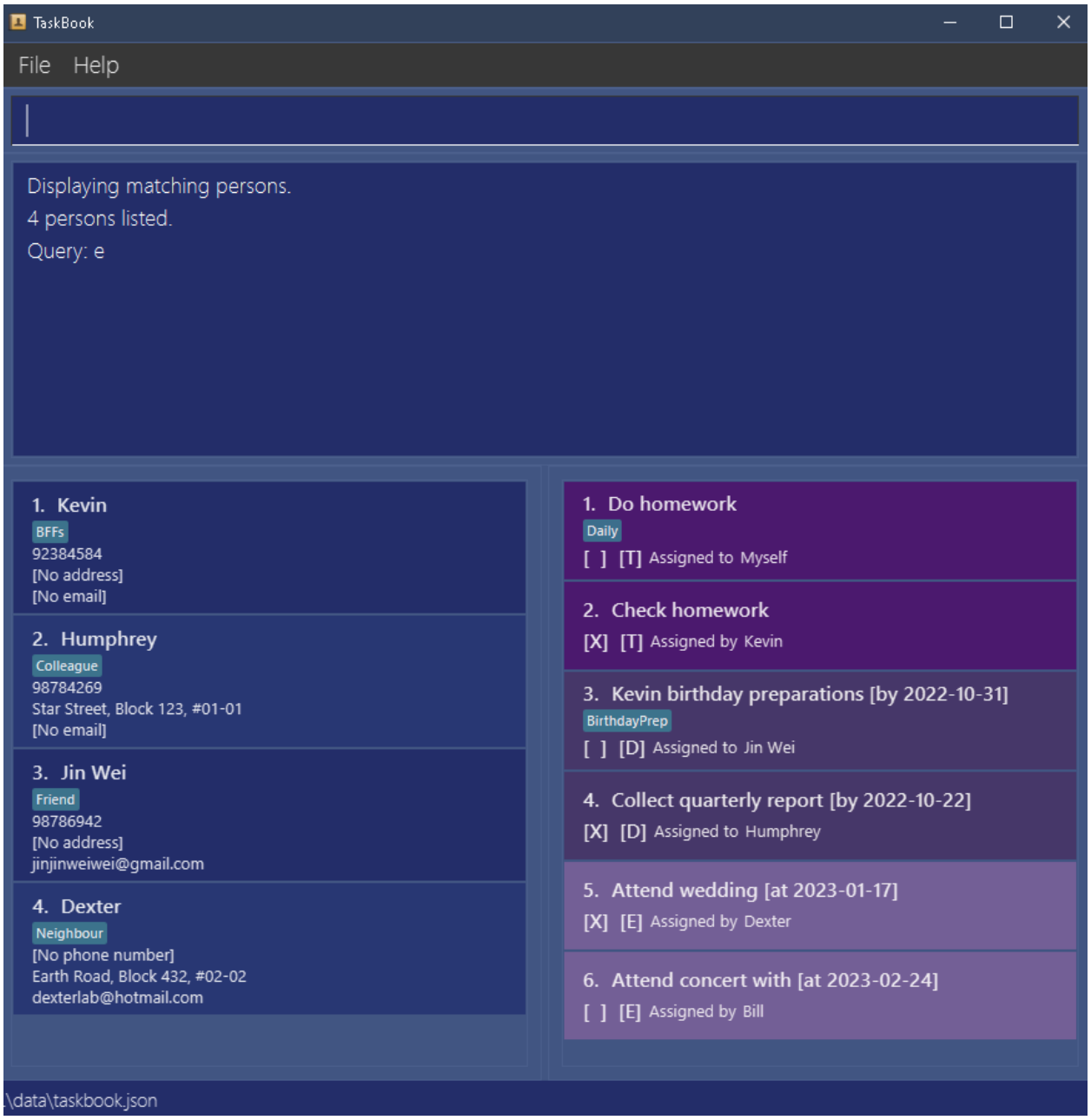
contact sort s/a



contact sort s/p

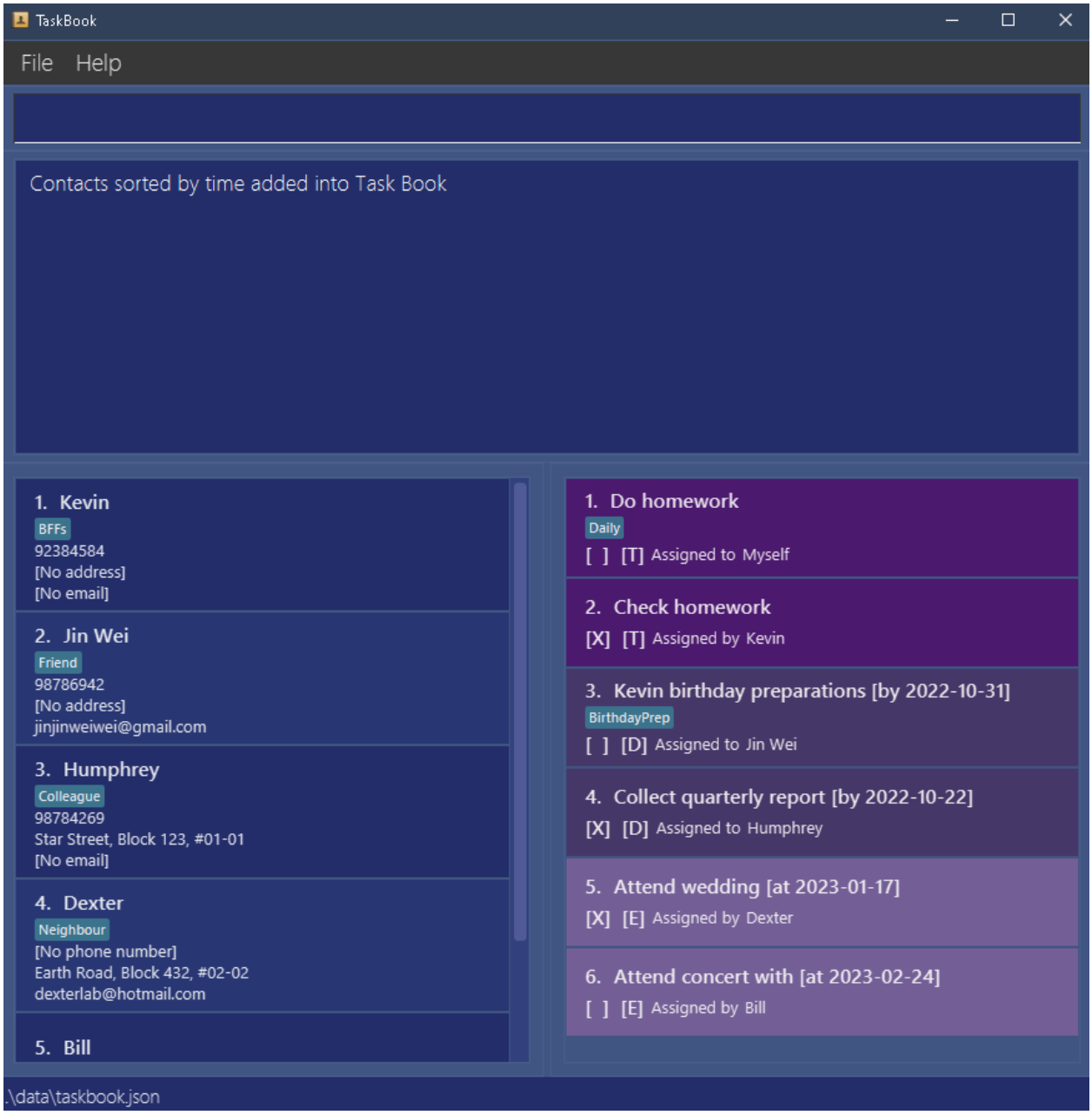


Special note: Filtering can be combined with sorting.
`contact find q/e` after `contact sort s/p`



`contact list` to reset filter

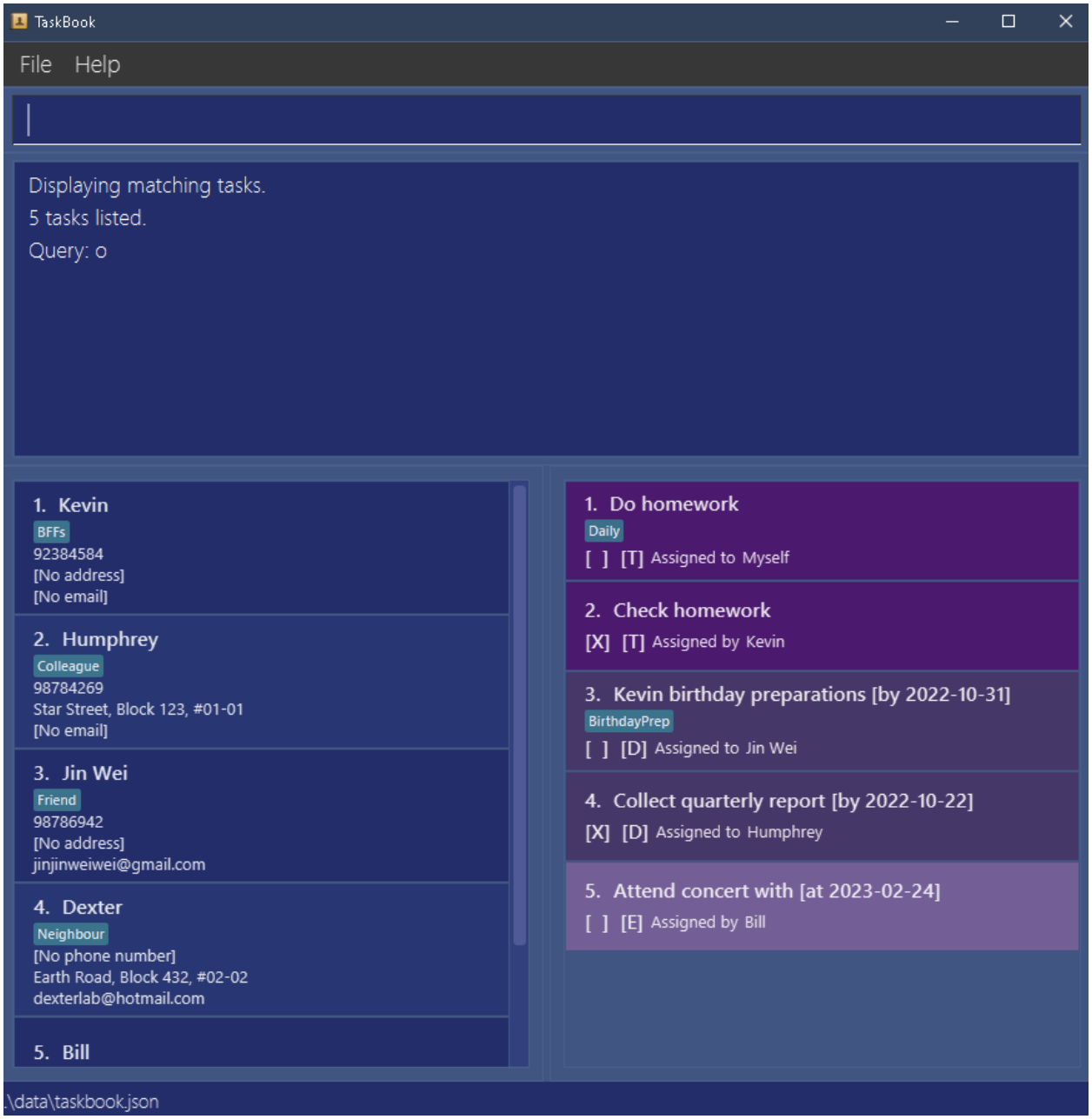
To reset the sort back to the original list without sorting,
contact sort s/ca
This also doubles as ‘sort by time added into Task Book’



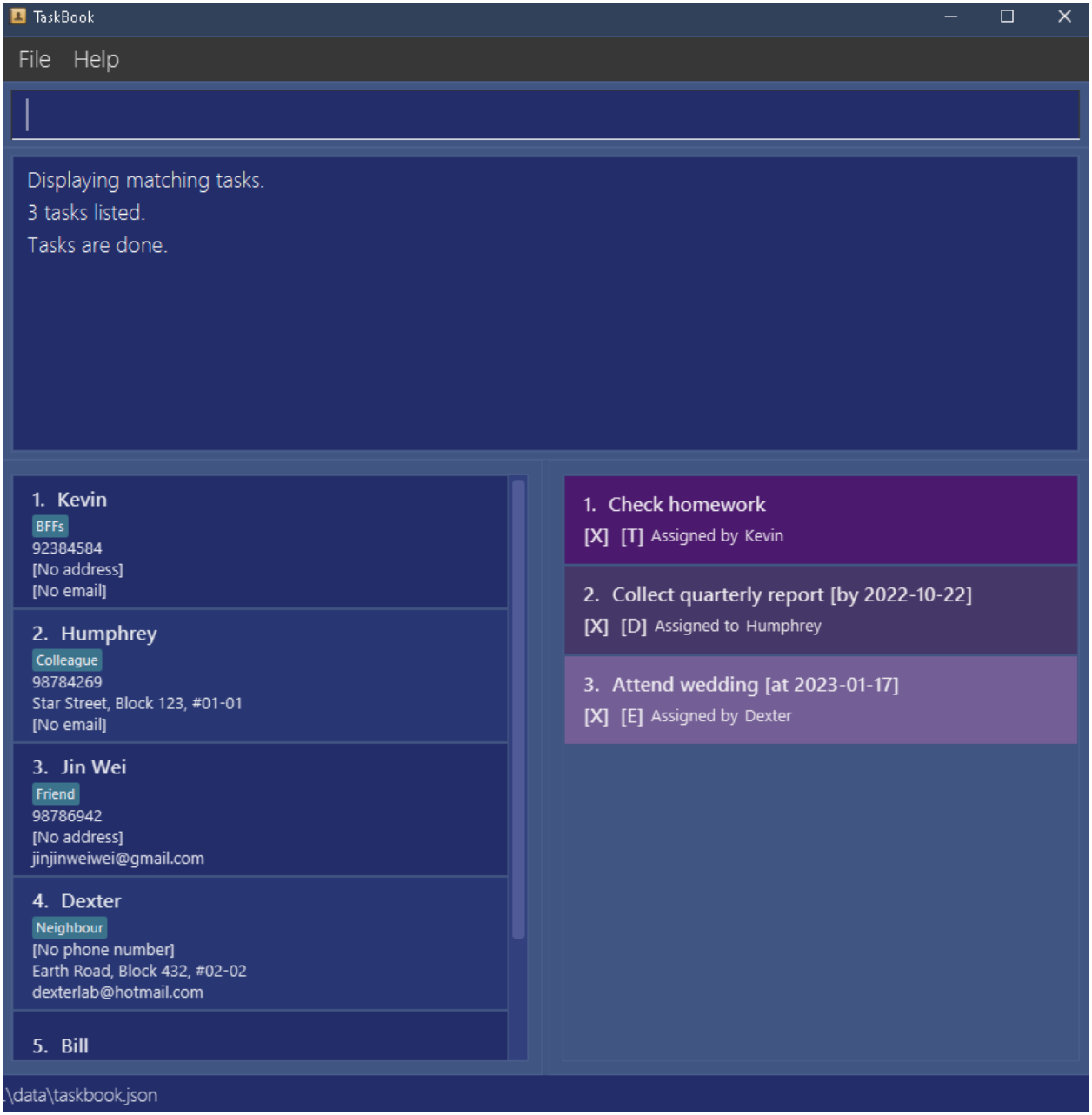
Task Organization

Finding Tasks

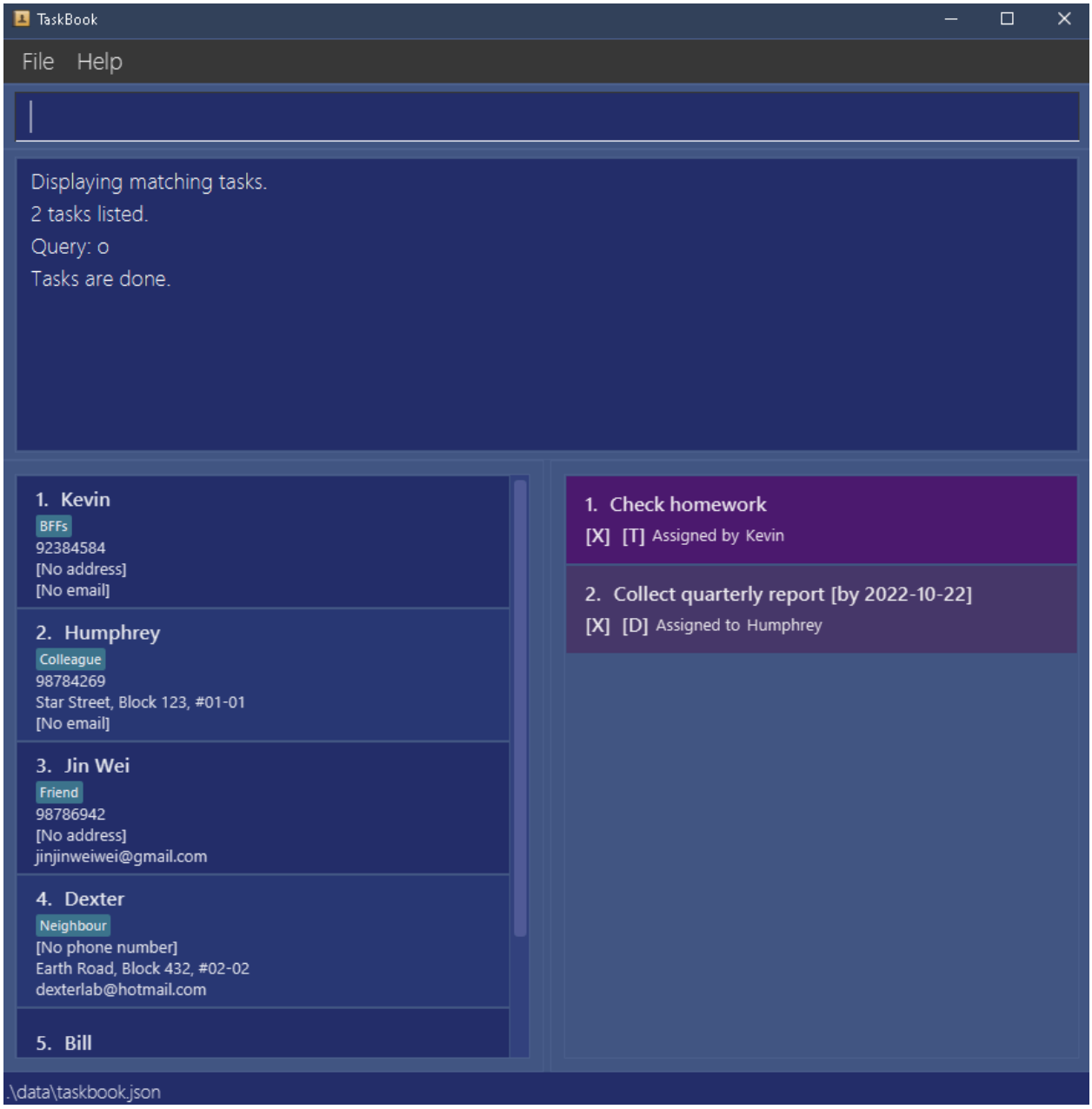
`task find` has 3 fields, and the user must supply at least one.
task find q/o



task find x/X

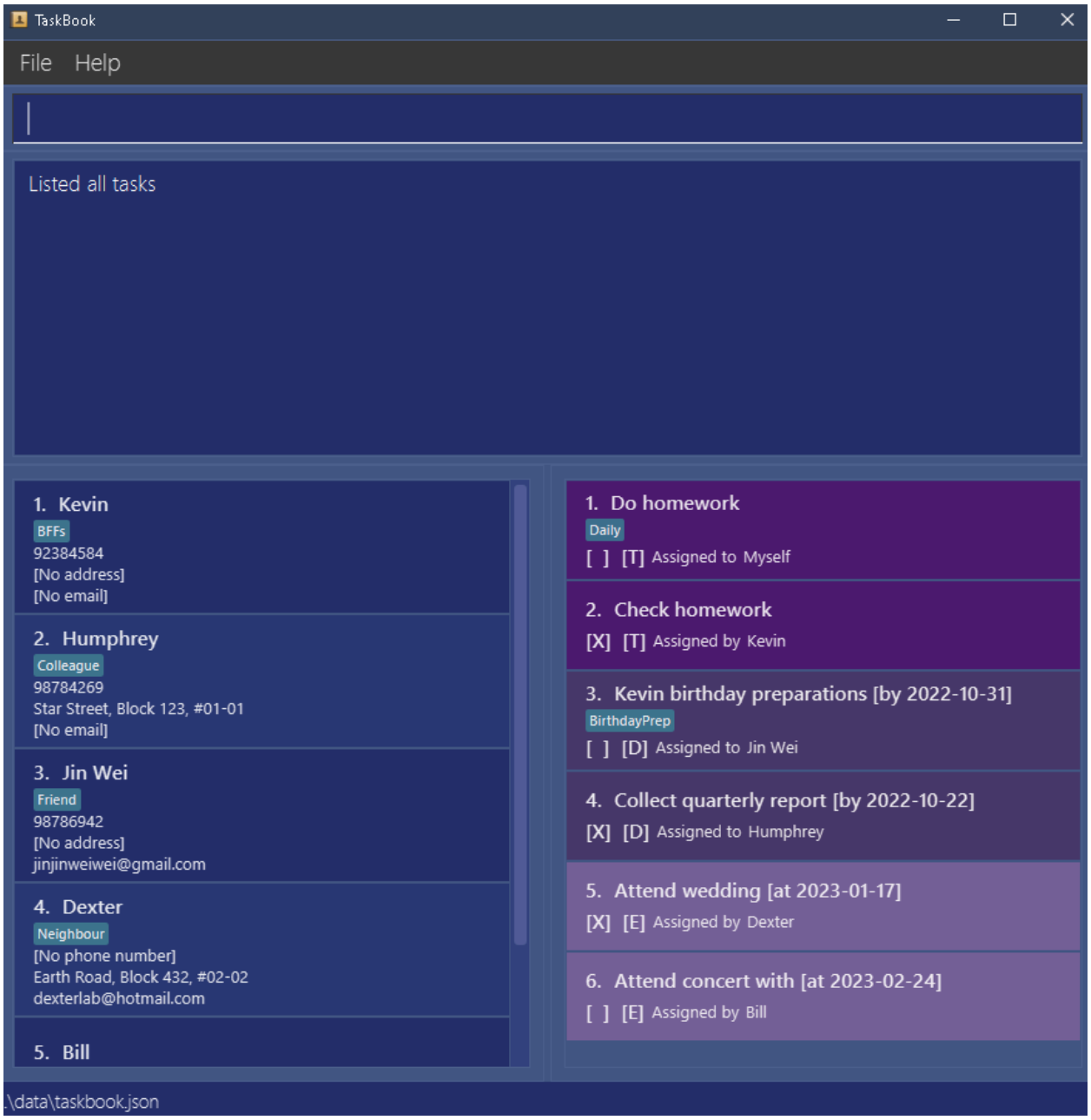

TaskBook
File
Help
Displaying matching tasks.
3 tasks listed.
Tasks are done.
1. Kevin
BFFs
92384584
[No address]
[No email]
2. Humphrey
Colleague
98784269
Star Street, Block 123, #01-01
[No email]
3. Jin Wei
Friend
98786942
[No address]
jinjinweiwei@gmail.com
4. Dexter
Neighbour
[No phone number]
Earth Road, Block 432, #02-02
dexterlab@hotmail.com
5. Bill
1. Check homework
[X] [T] Assigned by Kevin
2. Collect quarterly report [by 2022-10-22]
[X] [D] Assigned to Humphrey
3. Attend wedding [at 2023-01-17]
[X] [E] Assigned by Dexter
\data\taskbook.json

task find q/o x/X



Resetting Task Find Filters

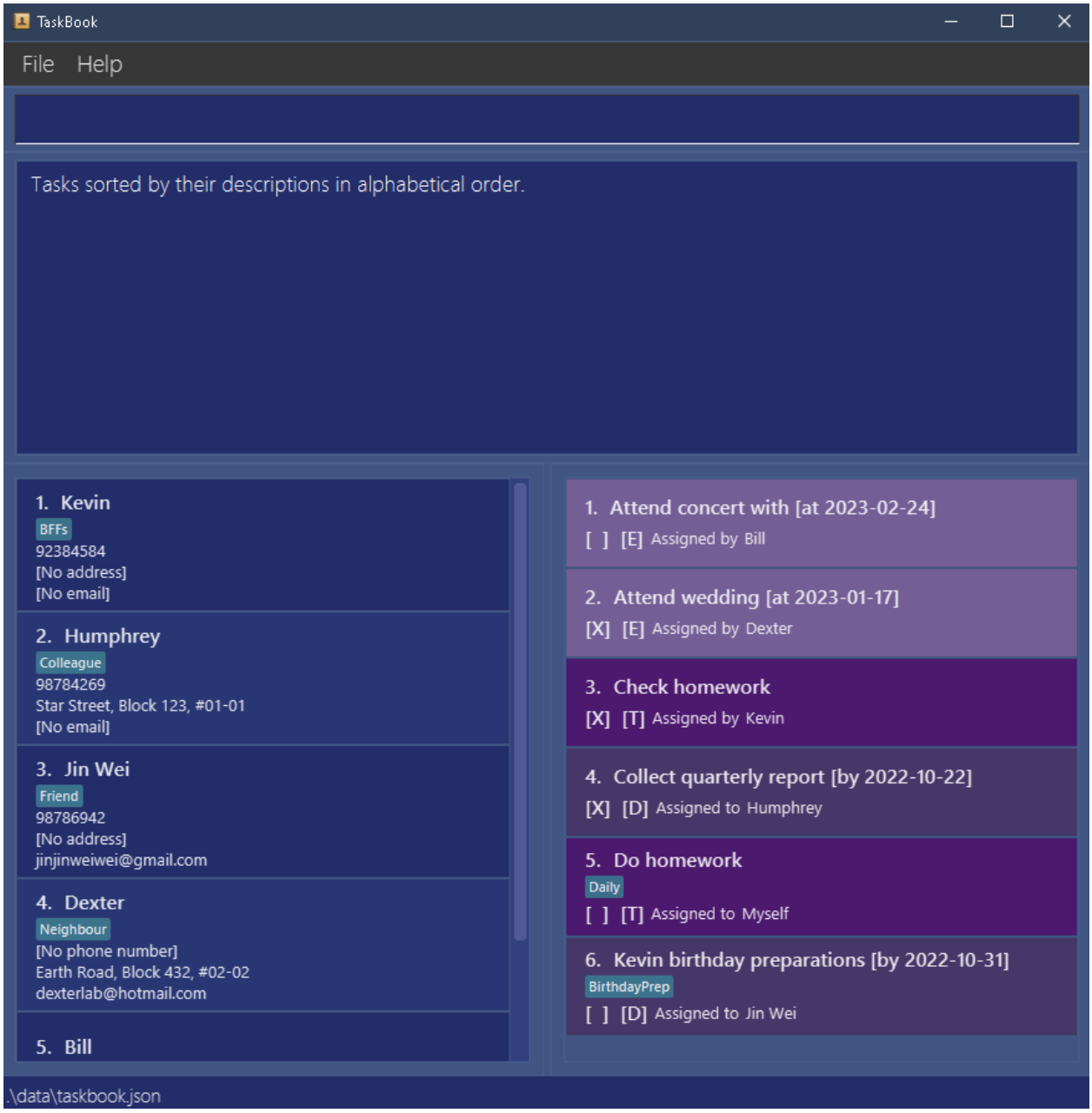
Similar to contact list, task list resets filters in task find.



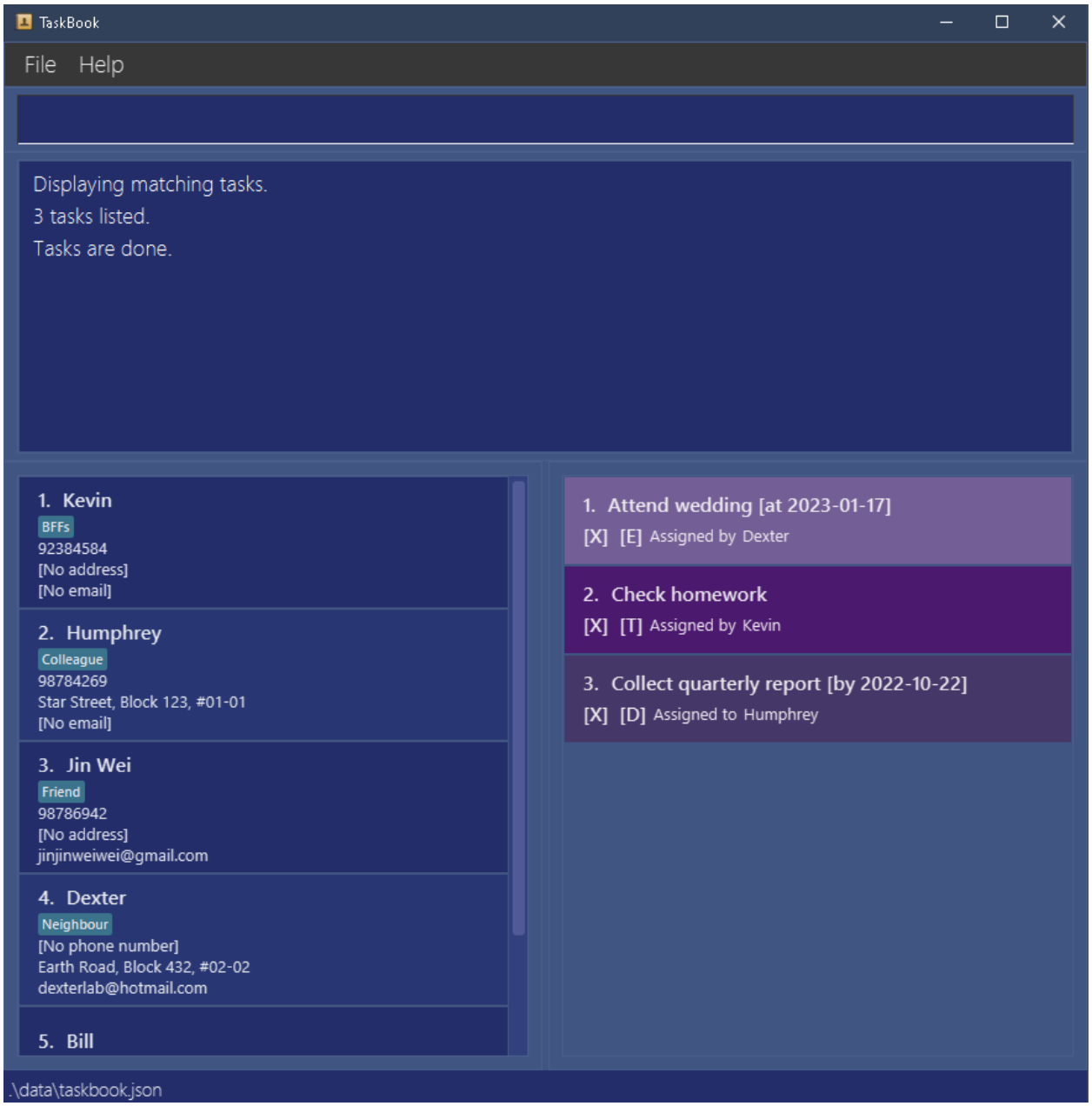
Sorting Tasks

Note: Operations on contact indices, i.e task delete, work on the index of the task that is being displayed on the GUI, not the actual original list. This is the more intuitive implementation.

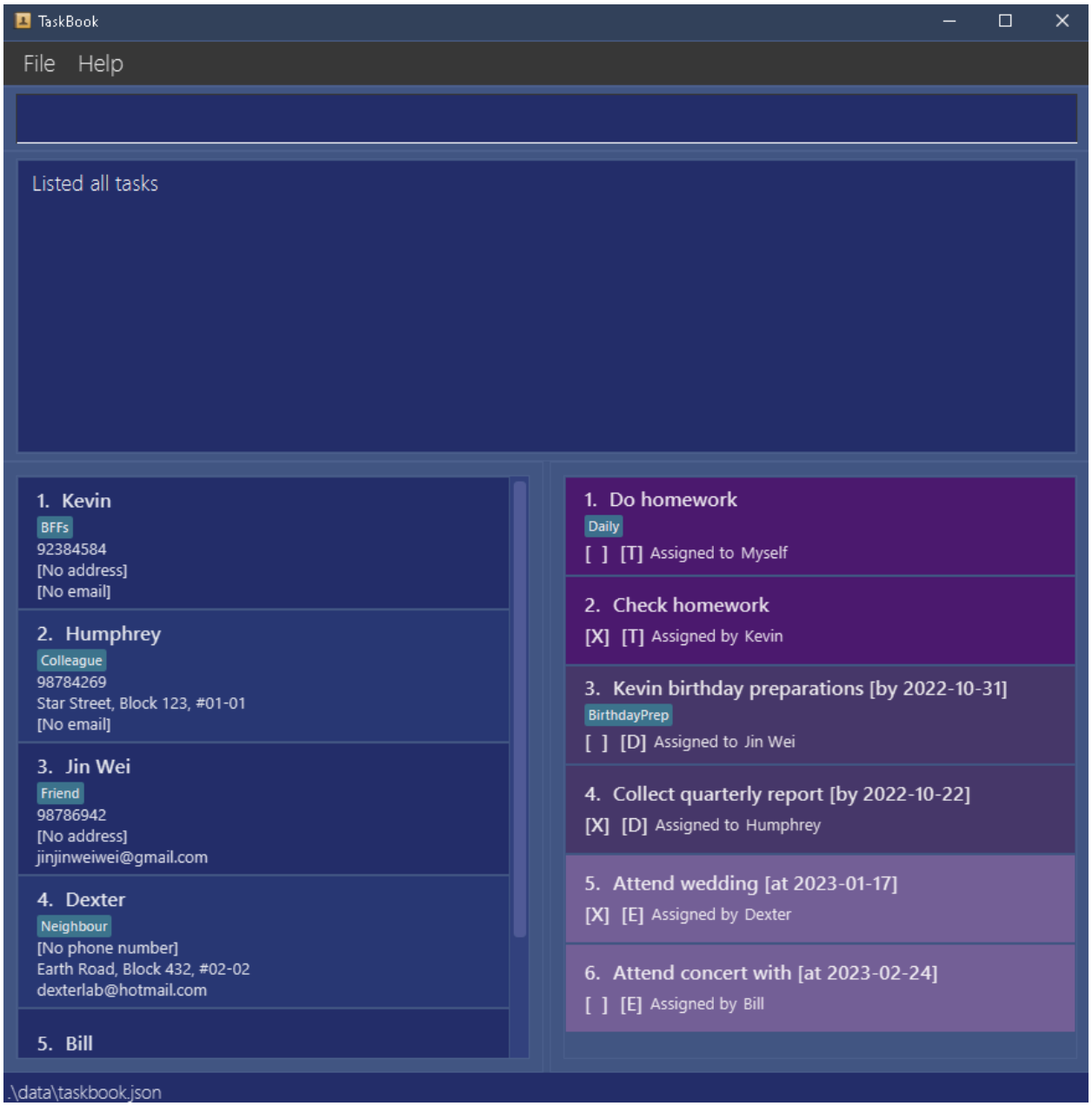
task sort s/a



Task sorts work with task find filters too
task find x/X



To reset the sort back to the original list without sorting,
contact sort s/ca
This also doubles as ‘sort by time added into Task Book’
After also entering in task list,

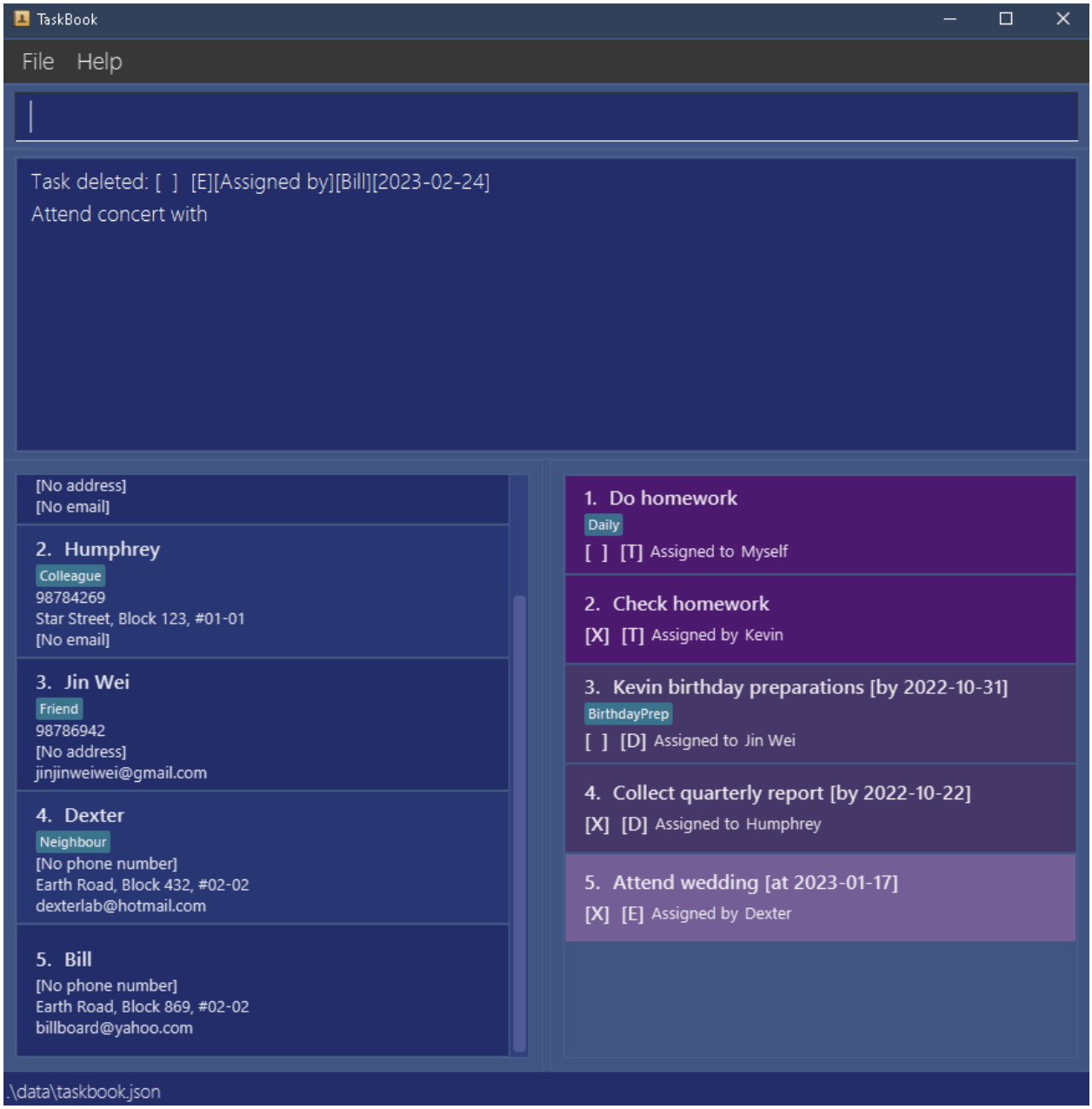


Task Book State Manipulation

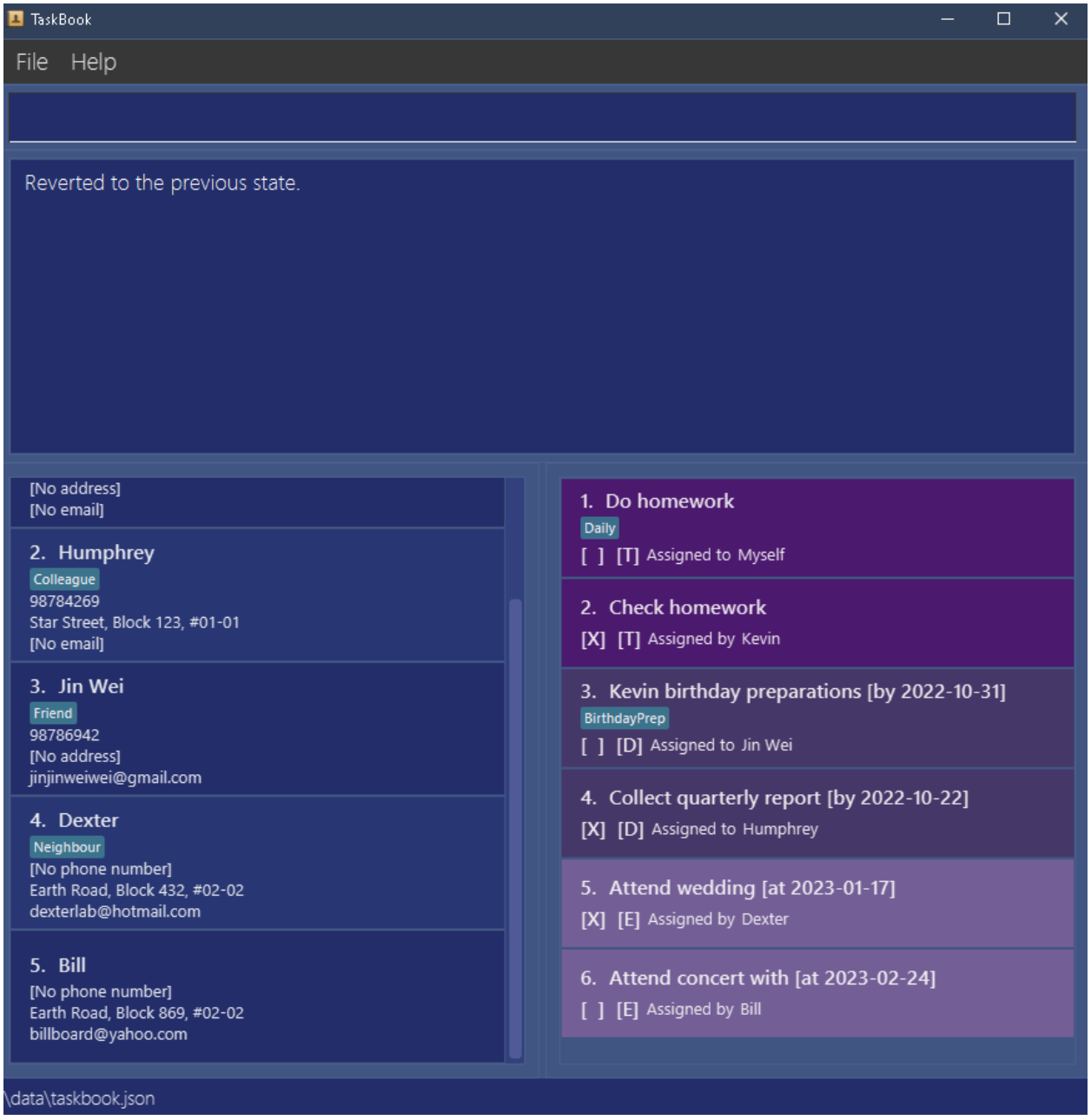
Undo

Oh no! We deleted a task by accident! Well, with the undo command, we can undo it!

task delete i/6



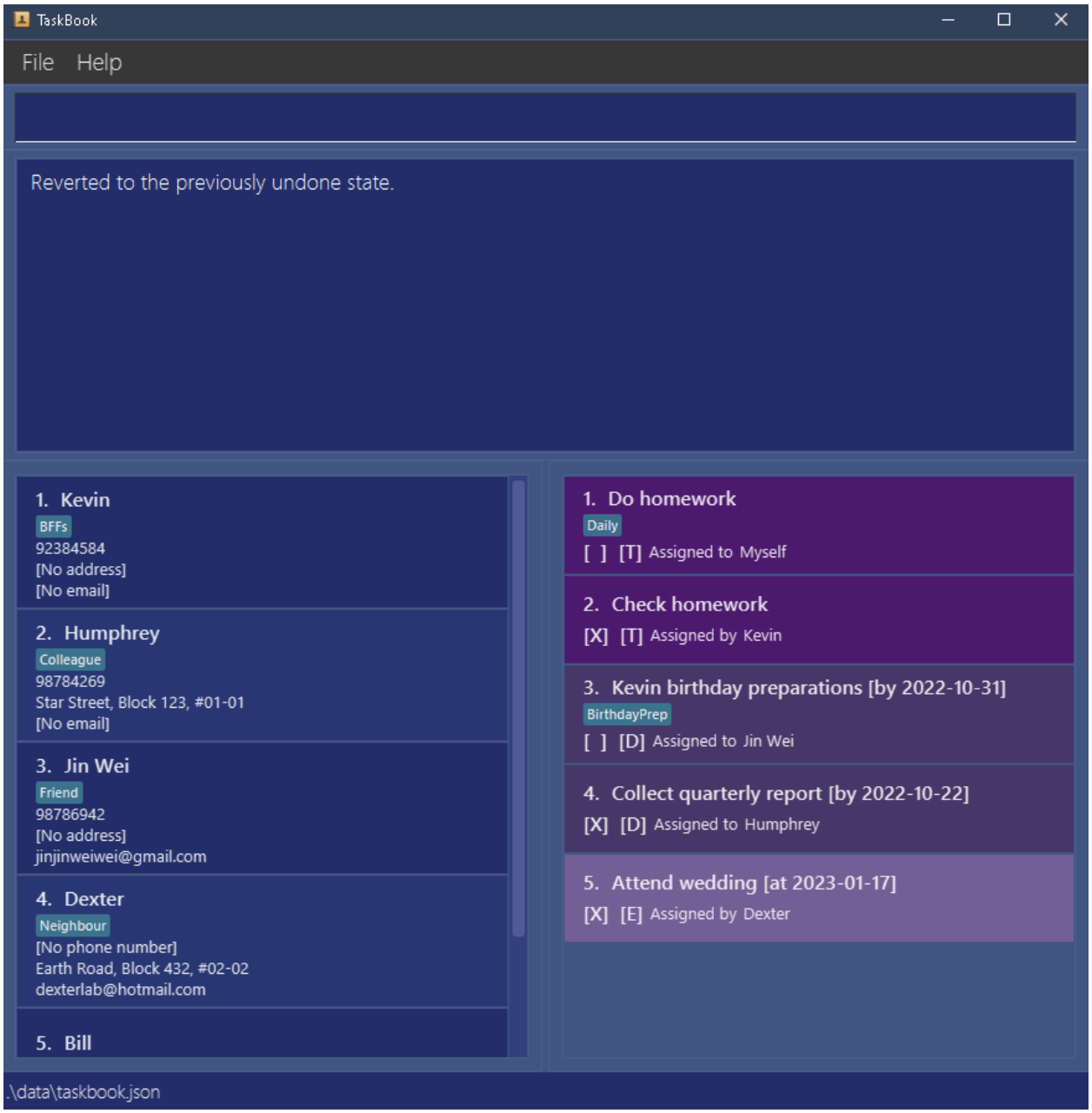
undo


TaskBook
File
Help
Reverted to the previous state.
[No address]
[No email]
2. Humphrey
Colleague
98784269
Star Street, Block 123, #01-01
[No email]
3. Jin Wei
Friend
98786942
[No address]
jinjinweiwei@gmail.com
4. Dexter
Neighbour
[No phone number]
Earth Road, Block 432, #02-02
dexterlab@hotmail.com
5. Bill
[No phone number]
Earth Road, Block 869, #02-02
billboard@yahoo.com
1. Do homework
Daily
[ ] [T] Assigned to Myself
2. Check homework
[X] [T] Assigned by Kevin
3. Kevin birthday preparations [by 2022-10-31]
BirthdayPrep
[ ] [D] Assigned to Jin Wei
4. Collect quarterly report [by 2022-10-22]
[X] [D] Assigned to Humphrey
5. Attend wedding [at 2023-01-17]
[X] [E] Assigned by Dexter
6. Attend concert with [at 2023-02-24]
[ ] [E] Assigned by Bill
\data\taskbook.json

Redo

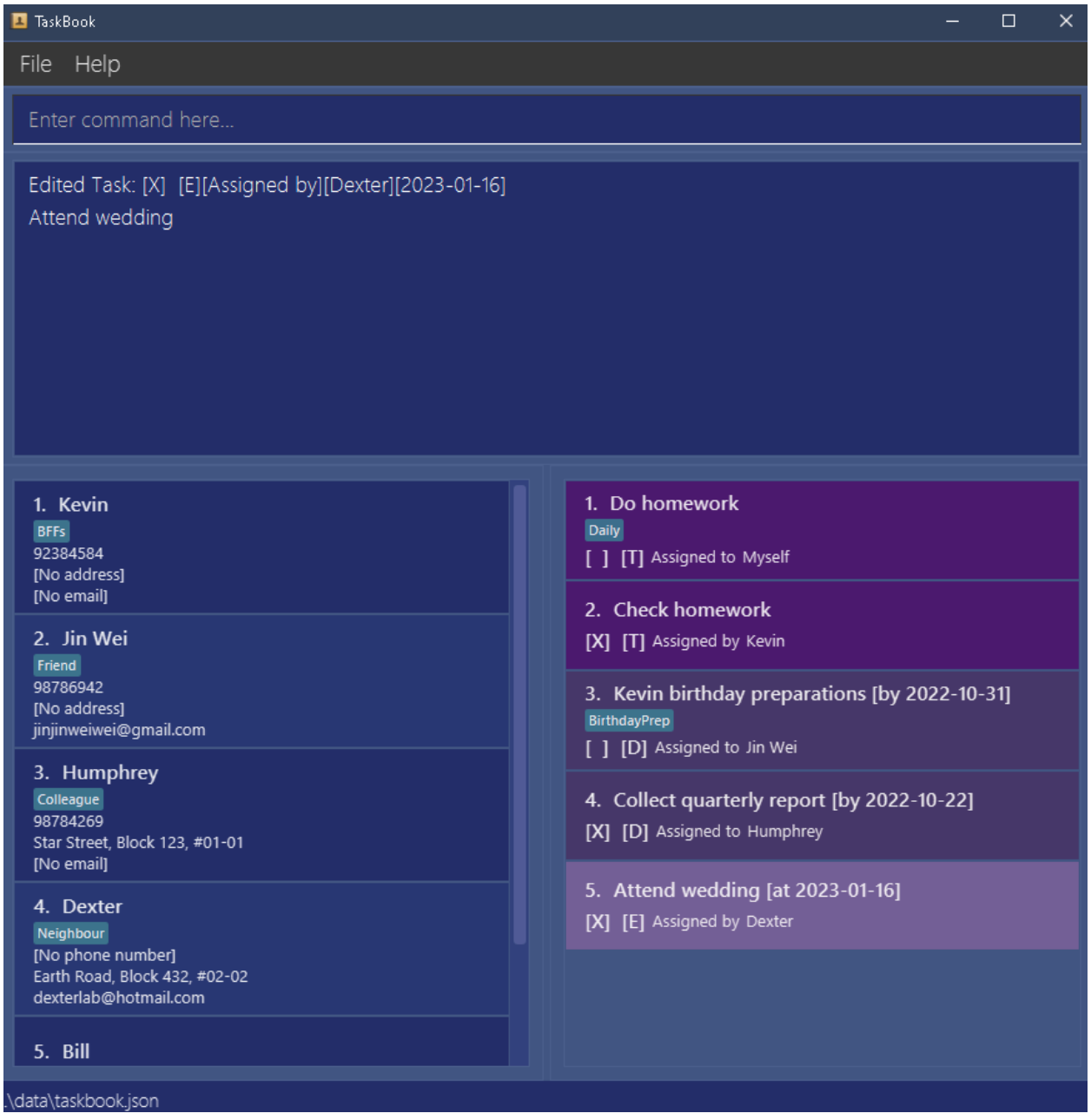
But what if you changed your mind again? What if you really want to delete that task?

redo



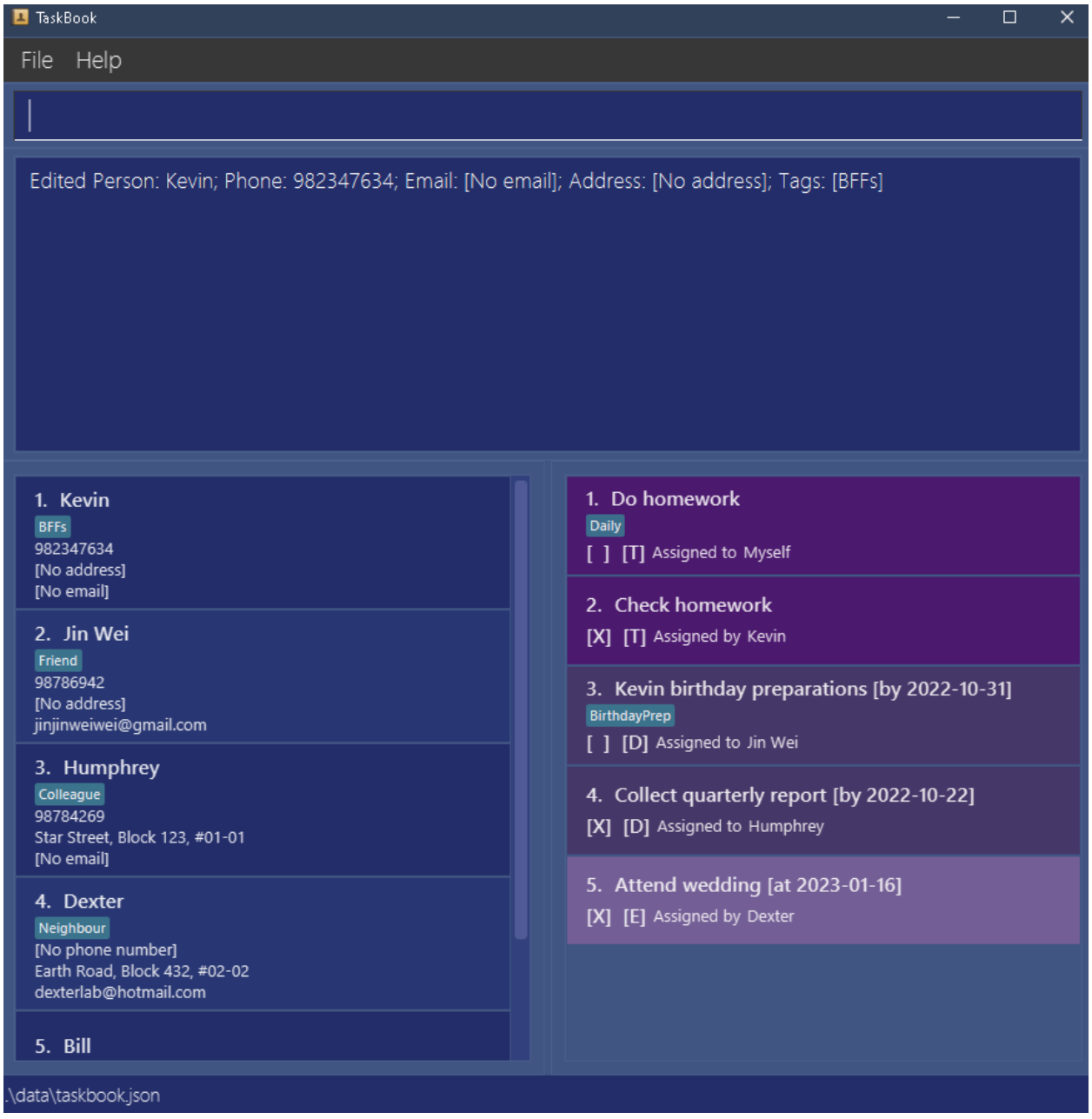
Editing Tasks

task edit i/5 m/Dexter d/Attend wedding t/2023-01-16



Editing Contacts

contact edit i/1 p/982347632



Exiting Task Book

bye exits the program immediately.
We can also click Exit under the File menu in the top left.

User Guide Link

The link to the user guide can be obtained by clicking the Help menu button in the top left.