Shreyas Kuthanoor Prakash - Project Portfolio

PROJECT: Police Records and Intelligence System (PRISM)

1. Overview

This document serves to showcase my contributions towards the CS2113T Software Engineering Project, as well as the role I served. My team and I have built PRISM using AddressBook - Level 3, provided by the se-edu team as a base.

The Police Records and Intelligence System (PRISM) is a desktop application catered towards the needs of the Police. The System has two types of users- Police Officers and Headquarters Personnel, each with different levels of access. Both types of users interact with the System using a CLI (Command-Line Interface). The application is written in Java, and it has a minimalistic GUI (Graphical User Interface) created with JavaFX.

The main features of the project are - Password, AutoCorrection, Screening History, Notifications and Editing by NRIC. Please refer to the Quick Start guide to get started.

2. Summary of contributions

This section summarizes my contributions towards the project and the role I played.

- **Role**: Developer

- **Responsibility**: Code Quality
- **Major enhancement**: **AutoCorrect functionality**
    - What it does: Provides suggestions when the user, either Police Officer or Headquarters Personnel, types in an incorrect input.
    - Justification: Human beings tend to make mistakes sometimes. This is especially true in the case of the Police force because they have to respond to critical situations by taking the least possible time. Thus, the AutoCorrect feature is extremely useful in cases where minute errors are made, as the application suggests the right implementation of the otherwise invalid input.
    - Highlights: This enhancement covers all commands as well as the unique NRICs of subjects stored in the System database. It will also cover all subjects to be added in the future. The implementation was challenging as it required changes to every command which uses NRIC as input.
    - Credits: The Dynamic Programming algorithm that checks the number of single character changes needed to convert one string to another was reused from- https://www.programcreek.com/2013/12/edit-distance-in-java/
- **Minor enhancement**: The user can view the current local time and date using the time command. This is useful especially for providing backup as well as meeting deadlines.
- **Code contributed**: Functional & Test code (https://nuscs2113-ay1819s1.github.io/dashboard/#=undefined&search=shreyaskp&sort=displayName&since=2018-09-12&until=2018-10-30&timeframe=day&reverse=false&repoSort=true)
- **Other contributions**:
    - Project management:
        - Ensured team maintains code quality, according to the Java Coding Standard (https://oss-generic.github.io/process/codingStandards/CodingStandard-Java.html)
        - Added implementation of CheckStyle plugin to fix code quality bugs: #235 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/235), #242 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/242)
    - Documentation:
        - Added user stories relevant to our project in the Developer Guide: #4 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/4)
        - Added use cases for features implemented in the Developer Guide: #111 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/111)
        - Cleaned up Developer Guide: #111 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/111), #126 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/126)
    - Community:

- PRs reviewed (with non-trivial review comments): #115 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/115), #110 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/110), #130 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/130), #93 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/93), #98 (https://github.com/CS2113-AY1819S1-F10-3/main/pull/98)

3. Contributions to the User Guide

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users. For the User Guide, I added information on the AutoCorrect functionality.

3.1. AutoCorrect feature (HQP & PO)

Predicts expected input when the user enters invalid input that is close to input expected by the Police Records System. Currently all non-password commands and NRICs are covered by the feature.

- The feature predicts correction for commands that are a single character away from valid command
- The feature predicts correction for NRICs that are one or two characters away from valid NRIC

```
The feature works differently depending on the type of user.
```

- For HQP, all commands as well as NRICs under edit, check and delete command are covered by the feature.
- For POs, only commands they are authorized to use are covered by the feature.
- The autocorrect feature does not cover the update password command, and NRICs under commands accessible by POs to maintain security.

Examples:

- lost
- Predicts that the user meant to use the command `list` and shows valid implementation of the list command

3.2. Display date and time : `time` (HQP & PO)

Shows the current System date and time.

Format: 'time'

Examples:

- time
- Displays the current date, and time in hrs

4. Contributions to the Developer Guide

Given below are sections I contributed to the Developer Guide. They showcase my ability to write technical documentation and the technical depth of my contributions to the project. For the Developer Guide, I added the user stories, implementation and use cases of AutoCorrect feature, non functional requirements and helped clean the whole document.

Appendix B: User Stories

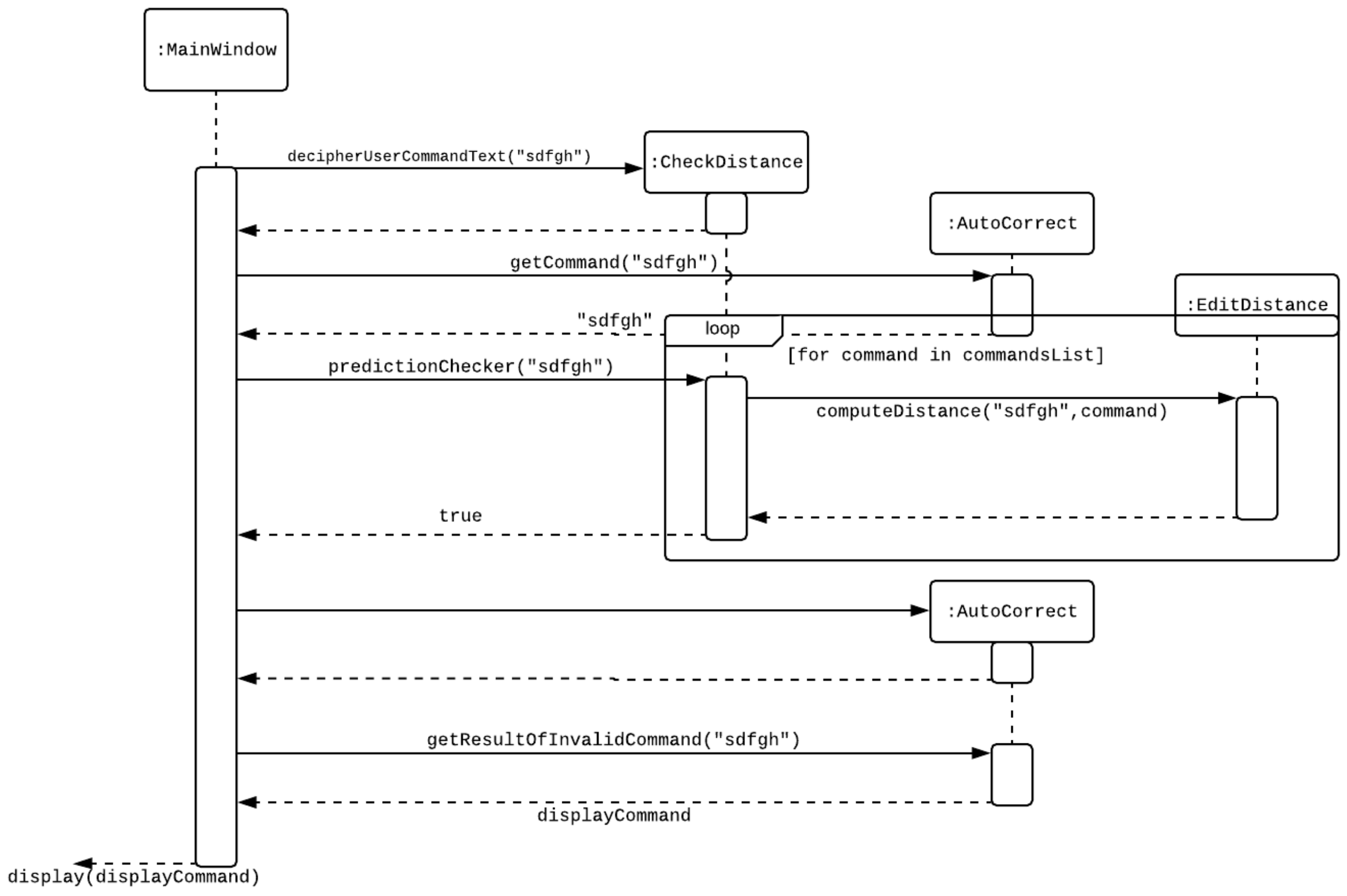
Priorities: High (must have) - * * *, Medium (nice to have) - * *, Low (unlikely to have) - *

PO- Police Officer HQP- Headquarters Personnel

Priority	As a …	I want to …	So that I can…
* * *	PO	request backup efficiently and quickly	get help in dangerous situations like capturing an escaped criminal, saving a person’s life
* * *	PO	know if accused is dangerous	know the steps I should take to handle the accused
* * *	PO	easily access numerous NRICs and commands with autocorrection	be efficient in going through many records even if some mistake is made
* * *	PO	quickly screen the subject using his NRIC	know his current status and past offences if any

Priority	As a …	I want to …	So that I can…
* * *	HQP	know the screening history of a particular subject using his NRIC	use it in my investigation
* * *	PO	secure my device with a password	prevent breach of confidential data
* *	HQP	update password of any device regularly	so that I can increase security
*	PO	know the serial number and battery level	to return it to HQ and charge it when necessary

Appendix A: 6. Autocorrection feature


:MainWindow
decipherUserCommandText("sdfgh")
:CheckDistance
:AutoCorrect
getCommand("sdfgh")
:EditDistance
"sdfgh"
loop
[for command in commandsList]
predictionChecker("sdfgh")
computeDistance("sdfgh",command)
true
:AutoCorrect
getResultOfInvalidCommand("sdfgh")
displayCommand
display(displayCommand)

Current Implementation

The autocorrect mechanism is facilitated by use of dynamic programming. The algorithm called EditDistance checks the number of single character changes to be made to convert an invalid input into one expected by the system. Currently, changes involving one single character can be corrected by the system for commands and changes involving one or two single characters can be corrected by the system for NRICs. It implements the following operations:

1. checkDistance() - It returns the edit distance needed to convert one string to the other. In this case, it returns the number of single character changes (either addition of a character, deletion of a character or changing a character) to convert invalid user input into its most probable correct implementation.

The following is an example usage scenario of the autocorrection feature for commands:

Step 1: The user inputs his password and unlocks the system.

Step 2: The user enters an invalid command.

Step 3: The system predicts the most probable intended command the user would have wanted to input and then prompts the user to use the prediction given in its valid format.

The following is an example usage scenario of the autocorrection feature for NRICs:

Step 1: The user inputs his password and unlocks the system.

Step 2: The user tries to edit, delete or check an invalid NRIC.

Step 3: The system predicts the most probable intended NRIC the user would have wanted to input and then prompts the user to use the prediction given in the valid format of the command.

The input is checked by the algorithm in the MainWindow before it is sent to the Parser class. This is to ensure invalid input can be caught by the algorithm to give its correction before it is deemed as invalid by the Parser during which time all commands will be laid out to the user.

B.A.1. Design Considerations

Aspect: How autocorrect feature is implemented

- **Alternative 1 (current choice):** Using user input directly from MainWindow class
  - Pros: Able to autocorrect raw input
  - Cons: More front-end coding through MainWindow class
- **Alternative 2:** Running the algorithm from the Parser class
  - Pros: Implementing new code with existing code
  - Cons: Difficult to run autocorrect through parser

B.1. Use case: Autocorrection of commands

MSS

1. User opens System.
2. System prompts user to enter his password.
3. User enters password.
4. System prompts user to enter his command.
5. User enters invalid command.
6. System predicts what command the user would have wanted to type if it finds a correction and displays the valid implementation of the command.

   Use case ends.

Extensions

- 3a. User enters an invalid password.
  - 3a1. System shows an error message.

    Use case resumes at step 2.
- 6a. User enters an invalid command for which the system cannot find a prediction.
  - 6a1. System shows error message.

    Use case ends.

B.2. Use case: Autocorrection of NRICs

MSS

1. User opens System.
2. System prompts user to enter his password.
3. User enters password.
4. System prompts user to enter his command.
5. User enters delete, edit or check command with invalid NRIC.
6. System predicts the NRIC the user would have wanted to type if it finds a correction and displays the valid implementation of the command.

   Use case ends.

Extensions

- 3a. User enters an invalid password.
  - 3a1. System shows an error message.

Use case resumes at step 2.

- 5a. User enters an invalid delete, edit or check command.
  - 5a1. System displays the valid usage of the command.

    Use case ends.
- 6a. User enters an invalid NRIC for which the system cannot find a prediction.
  - 6a1. System shows error message.

    Use case ends.

Last updated 2018-11-12 23:34:08 SRET