# Design Specification
# JobSeeker
Version 2

Zihao Du
Senni Tan
Gengyun Wang
Wenzhi Wang
Lab 3 Group 3
Computing and Software Department, Mcmaster University
SFWR ENG 2XB3, Software Engineering Practice and Experience:
Binding Theory to Practice

April 7, 2020

# Revision Page

By virtue of submitting this document we electronically sign and date that the work being submitted by all the individuals in the group is their exclusive work as a group and we consent to make available the application developed through [CS] or [SE]-2XB3 project, the reports, presentations, and assignments (not including my name and student number) for future teaching purposes.

***First revision:***

Senni Tan — Edited the title page and created the contribution table.

Zihao Du — Added the attestation and consent in Revision.

***Second revision:***

Senni Tan — Edited the contribution table.

Zihao Du— Edited the contribution table.

Wang Wenzhi — Edit the contribution table.

Gengyun Wang — Edited the contribution table.

# Contribution Page

| Name | Role(s) | Contribution | Comments |
|---|---|---|---|
| Zihao Du | Designer<br>Researcher<br>Designer<br><br><br><br>Tester | Proposal Abstract and motivation<br>Database of jobs<br>SRS Functional requirement<br>Graphing algorithm implementation<br>Client module<br>Unit test for graphing algorithm | |
| Senni Tan | Designer<br><br><br><br>Tester | Proposal I/O<br>SRS Non-functional requirement<br>Sorting Algorithm Implementation<br>Unit test for sorting algorithm implementation | |
| Gengyun Wang | Designer<br><br><br><br>Tester | Proposal Prior Work<br>SRS Assumptions, Domain<br>Searching Algorithm Implementation<br>Unit test for searching algorithm implementation | |
| Wenzhi Wang | Designer<br><br><br><br>Tester | Proposal Reference page<br>SRS Maintenance and Development<br>Data processing implementation<br>Test and modify the client code implementation | |

# Executive Summary

# Contents

# 1 Description of Modules

# 2 Detailed description of interfaces

# Comparator Module

## Module

Comparable

## Uses

Job

## Syntax

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| CompareString | Job, Job | $\mathbb{Z}$ | |
| CompareOutlook | Job, Job | $\mathbb{Z}$ | |
| CompareNOC | Job, Job | $\mathbb{Z}$ | |
| CompareRegionS | Job, Job | $\mathbb{Z}$ | |

## Semantics

### Access Routine Semantics

CompareString(a, b):

- output: *out* := a.get_title.compareTo(b.get_title)
- exception: None

*// compareTo is a build in method to compare String in lexgraphical order.*

CompareOutlook(a, b):

- output: *out* := (a.get_outlook > b.get_outlook) $\Rightarrow$ 1 | (a.get_outlook < b.get_outlook) $\Rightarrow$ -1 | 0
- exception: None

CompareNOC(a, b):

- output: *out* := (a.get_noc(0) > b.get_noc(0)) ⇒ 1 | (a.get_noc(0) < b.get_noc(0)) ⇒ -1 | 0
- exception: None

CompareRegionS(a, b):

- output: *out* := a.get_regions.compareTo(b.get_regions)
- exception: None

*// compareTo is a build in method to compare String in lexgraphical order.*

# Sorting Module

## Module

Sorting

## Uses

Comparable

## Syntax

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| sortString | Seq of Job | | |
| sortOutlook | Seq of Job | | |
| sortNOC | Seq of Job | | |
| sortRegionS | Seq of Job | | |

## Semantics

### Access Routine Semantics

sortString(a):

- transition: sortString(a, 0, $|a|$-1)
- exception: None

sortOutlook(a):

- transition: sortOutlook(a, 0, $|a|$-1)
- exception: None

sortNOC(a):

- transition: sortNOC(a, 0, $|a|$-1)
- exception: None

sortRegionS(a):

- transition: sortRegionS(a, 0, $|a|$-1)
- exception: None

## Local Functions

exch: Seq of Job $\times \mathbb{Z} \times \mathbb{Z} \rightarrow$ None
exch$(a, i, j) \equiv$ exchange a[i] and a[j] in the array

sortString: Seq of Job $\times \mathbb{Z} \times \mathbb{Z} \rightarrow$ None
sortString$(a, lo, hi) \equiv$ (hi <= lo) $\Rightarrow$ return | sortString(a, lo, j-1) && sortString(a, j+1, hi) where j = partitionString(a, lo, hi)

partitionString: Seq of Job $\times \mathbb{Z} \times \mathbb{Z} \rightarrow$ None
partitionString$(a, lo, hi) \equiv$ partition on array $a$ using ComapreString, see detail in code

sortOutlook: Seq of Job $\times \mathbb{Z} \times \mathbb{Z} \rightarrow$ None
sortOutlook$(a, lo, hi) \equiv$ (hi <= lo) $\Rightarrow$ return | sortOutlook(a, lo, j-1) && sortOutlook(a, j+1, hi) where j = partitionOutlook(a, lo, hi)

partitionOutlook: Seq of Job $\times \mathbb{Z} \times \mathbb{Z} \rightarrow$ None
partitionOutlook$(a, lo, hi) \equiv$ partition on array $a$ using ComapreOutlook, see detail in code

sortNOC: Seq of Job $\times \mathbb{Z} \times \mathbb{Z} \rightarrow$ None
sortNOC$(a, lo, hi) \equiv$ (hi <= lo) $\Rightarrow$ return | sortNOC(a, lo, j-1) && sortNOC(a, j+1, hi) where j = partitionNOC(a, lo, hi)

partitionNOC: Seq of Job $\times \mathbb{Z} \times \mathbb{Z} \rightarrow$ None
partitionNOC$(a, lo, hi) \equiv$ partition on array $a$ using ComapreNOC, see detail in code

sortRegionS: Seq of Job $\times \mathbb{Z} \times \mathbb{Z} \rightarrow$ None
sortRegionS$(a, lo, hi) \equiv$ (hi <= lo) $\Rightarrow$ return | sortRegionS(a, lo, j-1) && sortRegionS(a, j+1, hi) where j = partitionRegionS(a, lo, hi)

partitionRegionS: Seq of Job $\times \mathbb{Z} \times \mathbb{Z} \rightarrow$ None
partitionRegionS$(a, lo, hi) \equiv$ partition on array $a$ using ComapreRegionS, see detail in code

## 3 View of uses relationship

## 4 Trace back to requirements

## 5 Description of implementation

## 6 Internal review