Module Interface Specification for Park'd

Team #29, caPstOneGroup
Albert Zhou
Almen Ng
David Yao
Gary Gong
Jonathan Yapeter
Kabishan Suvendran

January 18, 2023

1 Revision History

Date	Version	Notes
Jan 18, 2023	1.0	Revision 0

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at Park'd Software Requirements Specification.

Symbol	Description
Park'd	Parking Lot Application
MIS	Module Interface Specification
MG	Module Guide
SRS	Software Requirements Specifications

Contents

3 Introduction

The following document details the Module Interface Specifications for Park'd, our parking assistant application. Park'd aims to help drivers find parking spaces by using machine learning algorithms to locate empty spaces from overhead cameras. Our application then directs drivers to those spaces, taking into account restrictions like reserved or accessible spaces. It will maintain a database of spaces as well as a navigation layout for a given parking lot.

Complementary documents include the SRS and MG documents. The full documentation and implementation can be found at Module Interface Specification and Module Guide documents, respectively.

4 Notation

The structure of the MIS for modules comes from *Software Design, Automated Testing, and Maintenance: A Practical Approach* Hoffman and Strooper (1995), with the addition that template modules have been adapted from *Fundamentals of Software Engineering* Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of *Software Design, Automated Testing, and Maintenance: A Practical Approach* Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Park'd.

Data Type	**Notation**	**Description**
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
null	ϵ	empty value
Boolean	$\mathbb{B}$	true or false
String	String	a sequence of characters
Seq	Seq	an ordered collection of elements
exists	$\exists$	true if there exists an element that satisfies a property, false otherwise
for all	$\forall$	true if all elements satisfy a property, false otherwise
implies	$\Rightarrow$	true if the left operator is true then output the right operator, false otherwise
in	$\in$	true if a an element is in a Seq
and	$\wedge$	true if both operators are true, false otherwise
subset	$\subseteq$	true if a set contains another, false otherwise

The specification of Park'd uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, Park'd uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding Module	Camera capture module
Behaviour-Hiding Module	Admin console module
	Admin module
	Parking lot layout module
	Parking layout element module
	Parking spot module
	Authentication module
	User module
	User action handler module
	Vehicle module
	View module
Software Decision Module	Navigation module
	Parking Stats module
	Machine learning model module

Table 1: Module Hierarchy

6 MIS of Authentication Module

6.1 Module

AuthT

6.2 Uses

UserT

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Types

AuthT = ?

6.3.3 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
authenticateUser	String, String	UserT	MissingUserException

6.4 Semantics

6.4.1 State Variables

None

6.4.2 Environment Variables

$users$: Seq of UserT

6.4.3 Assumptions

None

6.4.4 Access Routine Semantics

authenticateUser(id, $pass$):

- output: $out := \exists(i : \mathbb{N} | i < |users| \wedge users[i].getUserId() = id \wedge users[i].getPassword() = pass)) \Rightarrow users[i]$

- exception: $exc := \neg\exists(i : \mathbb{N} | i < |users| \wedge users[i].getUserId() = id \wedge users[i].getPassword() = pass)) \Rightarrow MissingUserException$

7 MIS of User Module

7.1 Template Module

UserT

7.2 Uses

VehicleT

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Types

UserT = ?

7.3.3 Exported Access Programs

Name	In	Out	Exceptions
new UserT	String, String, VehicleT	UserT	UserCreationException
getUserId		String	
getPassword		String	
getVehicle		VehicleT	

7.4 Semantics

7.4.1 State Variables

userId: String
password: String
vehicle: VehicleT

7.4.2 Environment Variables

None

7.4.3 Assumptions

None

7.4.4 Access Routine Semantics

new UserT(id, $pass$, veh):

- transition: $userId$, $password$, $vehicle$:= id, $pass$, veh
- output: out := self
- exception: exc := $((|id| = 0) \lor (|pass| = 0) \lor (veh = \epsilon) \Rightarrow$ UserCreationException)

getUserId():

- output: out := $userId$
- exception: none

getPassword():

- output: out := $password$
- exception: none

getVehicle():

- output: out := $vehicle$
- exception: none

8 MIS of Vehicle Module

8.1 Template Module

VehicleT

8.2 Uses

None

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Types

VehicleT = ?

8.3.3 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
new VehicleT	$\mathbb{R}$, $\mathbb{R}$		VehicleCreationException
getLength		$\mathbb{R}$	
getWidth		$\mathbb{R}$	

8.4 Semantics

8.4.1 State Variables

$length$: $\mathbb{R}$
$width$: $\mathbb{R}$

8.4.2 Environment Variables

None

8.4.3 Assumptions

None

8.4.4 Access Routine Semantics

new VehicleT(*len*, *wid*):

- transition: $length, width := len, wid$
- output: $out :=$ self
- exception: exc := $((len \leq 0) \vee (wid \leq 0) \Rightarrow$ VehicleCreationException)

getLength():

- output: $out := length$
- exception: none

getWidth():

- output: $out := width$
- exception: none

9 MIS of Admin Console Module

9.1 Module

AdminConsoleT

9.2 Uses

AdminT

9.3 Syntax

9.3.1 Exported Constants

None

9.3.2 Exported Types

AdminConsoleT = ?

9.3.3 Exported Access Programs

Name	In	Out	Exceptions
authenticateAdmin	String, String	AdminT	MissingAdminException

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Environment Variables

admins: Seq of AdminT

9.4.3 State Invariant

None

9.4.4 Assumptions

None

9.4.5 Access Routine Semantics

authenticateAdmin($id, pass$):

- output: $out := \exists(i : \mathbb{N} | i < |admins| \land admins[i].getAdminId() = id \land admins[i].getPassword() = pass)) \Rightarrow admins[i]$
- exception: $exc := \neg\exists(i : \mathbb{N} | i < |admins| \land admins[i].getAdminId() = id \land admins[i].getPassword() = pass)) \Rightarrow MissingAdminException$

10 MIS of Admin Module

10.1 Template Module

AdminT

10.2 Uses

ParkingLotLayoutT, ParkingLayoutElemT, ParkingSpotT

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Types

AdminT = ?

10.3.3 Exported Access Programs

Name	In	Out	Exceptions
new AdminT	String, String	AdminT	AdminCreationException
getAdminId		String	
getPassword		String	
getLayout	String	ParkingLotLayoutT	LayoutNotFoundException
changeLayout	String, ParkingLayoutElemT		LayoutNotFoundException

10.4 Semantics

10.4.1 State Variables

$adminId$: String
$password$: String
$layouts$: seq of ParkingLotLayoutT

10.4.2 Environment Variables

None

10.4.3 State Invariant

None

10.4.4 Assumptions

None

10.4.5 Access Routine Semantics

new AdminT($id, pass$):

- transition: $adminId, password, layouts := id, pass, \langle\rangle$
- output: $out := self$
- exception: $exc := ((|id| = 0) \lor (|pass| = 0) \Rightarrow AdminCreationException)$

getAdminId():

- output: $out := adminId$
- exception: none

getPassword():

- output: $out := password$
- exception: none

getLayout($layoutId$):

- output: $out := \exists(i : \mathbb{N}|i < n : layout[i].getLayoutId() = layoutId) \Rightarrow layout[i]$
- exception: $exc := \neg\exists(i : \mathbb{N}|i < n : layout[i].getLayoutId() = layoutId) \Rightarrow LayoutNotFoundException$

changeLayout($layoutId, newSpot$):

- transition: $layout := \langle i : \mathbb{N}|i < n : layout[i].getLayoutSpotId() = spotId \Rightarrow layout[i].changeElem(newSpot.getElemId(), newSpot)|true \Rightarrow layout[i]\rangle$
- exception: $exc := \neg\exists(i : \mathbb{N}|i < n : layout[i].getLayoutId() = layoutId) \Rightarrow LayoutNotFoundException$

11 MIS of Parking Lot Layout Module

11.1 Template Module

ParkingLotLayoutT

11.2 Uses

ParkingLayoutElemT

11.3 Syntax

11.3.1 Exported Constants

None

11.3.2 Exported Types

ParkingLotLayoutT = ?

11.3.3 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
new ParkingLotLayoutT	String, $\mathbb{N}$	ParkingLotLayoutT	LayoutCreationException
setAllRoads			
getLayoutId		String	
getSize		$\mathbb{N}$	
changeElem	String, ParkingLayoutElemT		
getLayout		Seq of ParkingLayoutElemT	
getElem	String	ParkingLayoutElemT	ElemNotFoundException
getElemIndex	String	$\mathbb{N}$	ElemNotFoundException

11.4 Semantics

11.4.1 State Variables

$layoutId$: String
n: $\mathbb{N}$
$layout$: seq of ParkingLayoutElemT

11.4.2 Environment Variables

None

11.4.3 State Invariant

$|layout| = n^2$.

11.4.4 Assumptions

None

11.4.5 Access Routine Semantics

new ParkingLotLayoutT($id, size$):

- transition: $layoutId, n, layout := id, size, \langle\rangle$
- output: $out := self$
- exception: exc := $((|id| = 0) \Rightarrow AdminCreationException)$

setAllRoads():

- output: $layout := i : \mathbb{N} | i < n^2 : layout || \langle newParkingLayoutElem("road" + i, "road) \rangle$
- exception: none

getLayoutId():

- output: $out := layoutId$
- exception: none

getSize():

- output: $out := n$
- exception: none

changeElem($elemId, newElem$):

- transition: $layout := \langle i : \mathbb{N} | i < n : layout[i].getElemId() = elemId \Rightarrow newElem | true \Rightarrow layout[i] \rangle$
- exception: none

getLayout():

- output: $out := layout$

- exception: none

getElem($elemId$):

- output: $out := \exists(i : \mathbb{N} | i < n : layout[i].getLayoutId() = elemId) \Rightarrow layout[i]$
- exception: $exc := \neg\exists(i : \mathbb{N} | i < n : layout[i].getLayoutId() = elemId) \Rightarrow ElemNotFoundException$

getElemIndex($elemId$):

- output: $out := \exists(i : \mathbb{N} | i < n : layout[i].getLayoutId() = elemId) \Rightarrow i$
- exception: $exc := \neg\exists(i : \mathbb{N} | i < n : layout[i].getLayoutId() = elemId) \Rightarrow ElemNotFoundException$

12 MIS of Parking Layout Element Module

12.1 Template Module

ParkingElemT

12.2 Uses

None

12.3 Syntax

12.3.1 Exported Constants

None

12.3.2 Exported Types

ParkingElemT = ?

12.3.3 Exported Access Programs

Name	In	Out	Exceptions
new ParkingElemT	String, String	ParkingElemT	
getElemId		String	
getType		String	

12.4 Semantics

12.4.1 State Variables

elemId: String
type: String

12.4.2 Environment Variables

None

12.4.3 State Invariant

$type \in \{"spot", "road", "obstacle"\}$.

12.4.4 Assumptions

None

12.4.5 Access Routine Semantics

new ParkingElem(id, s, x, y):

- transition: $elemId, type := id, s$
- output: $out := self$
- exception: exc := None

getElemId():

- output: $out := elemId$
- exception: none

getType():

- output: $out := type$
- exception: none

13 MIS of Parking Spot Module

13.1 Template Module Inherits Parking Layout Element

ParkingSpotT

13.2 Uses

ParkingLayoutElemT

13.3 Syntax

13.3.1 Exported Constants

None

13.3.2 Exported Types

ParkingSpotT = ?

13.3.3 Exported Access Programs

Name	In	Out	Exceptions
new ParkingSpotT	String, $\mathbb{R}$, $\mathbb{R}$	ParkingSpotT	
setEnabled	$\mathbb{B}$		
setOccupied	$\mathbb{B}$		
setReserved	$\mathbb{B}$		
setHandicapped	$\mathbb{B}$		
setAllProp	Seq of $\mathbb{B}$		InvalidParamException
getEnabled		$\mathbb{B}$	
getOccupied		$\mathbb{B}$	
getReserved		$\mathbb{B}$	
getHandicapped		$\mathbb{B}$	
getAllProp		Seq of $\mathbb{B}$	

13.4 Semantics

13.4.1 State Variables

$enabled$: $\mathbb{B}$
$occupied$: $\mathbb{B}$
$reserved$: $\mathbb{B}$

handicapped: $\mathbb{B}$

13.4.2 Environment Variables

None

13.4.3 State Invariant

None

13.4.4 Assumptions

None

13.4.5 Access Routine Semantics

new ParkingSpot(id):

- transition: $spotId, type, enabled, reserved, handicapped, occupied := id, "spot", true, false, false, false$
- output: $out := self$
- exception: exc := None

setEnabled(e):

- transition: $enabled := e$
- exception: none

setOccupied(o):

- transition: $occupied := o$
- exception: none

setReserved(r):

- transition: $reserved := r$
- exception: none

setHandicapped(h):

- transition: $handicapped := h$
- exception: none

setAllProp(p):

- transition: $enabled, occupied, reserved, handicapped := p[0], p[1], p[2], p[3]$
- exception: $exc := |p| \neq 4 \Rightarrow InvalidParamException$

getEnabled():

- output: $out := enabled$
- exception: none

getOccupied():

- output: $out := occupied$
- exception: none

getReserved():

- output: $out := reserved$
- exception: none

getHandicapped():

- output: $out := handicapped$
- exception: none

getAllProp():

- output: $out := \langle enabled, occupied, reserved, handicapped \rangle$
- exception: none

14 MIS of Parking Stats Module

14.1 Module

ParkingStatsT

14.2 Uses

ParkingLotLayoutT, ParkingSpotT

14.3 Syntax

14.3.1 Exported Constants

None

14.3.2 Exported Types

ParkingStatsT = ?

14.3.3 Exported Access Programs

Name	In	Out	Exceptions
getStat	ParkingLotLayoutT, Seq of $\mathbb{B}$	$\mathbb{N}$	InvalidParamException

14.4 Semantics

14.4.1 State Variables

None

14.4.2 Environment Variables

None

14.4.3 State Invariant

None

14.4.4 Assumptions

None

14.4.5 Access Routine Semantics

getStat(l, p):

- output: $out := +(\forall(e : ParkingSpotT | e \in l.getLayout() : e.getType() = \text{"}spot\text{"} \wedge e.getAllProp() = p \Rightarrow 1 | true \Rightarrow 0))$
- exception: $exc := |p| \neq 4 \Rightarrow InvalidParamException$

15 MIS of Navigation Module

15.1 Module

NavigationT

15.2 Uses

ParkingLotLayoutT, ParkingLayoutElemT

15.3 Syntax

15.3.1 Exported Constants

None

15.3.2 Exported Types

NavigationT = ?

15.3.3 Exported Access Programs

Name	In	Out	Exceptions
findPath	ParkingLotLayoutT, String, String	Seq of String	NoPathException

15.4 Semantics

15.4.1 State Variables

None

15.4.2 Environment Variables

None

15.4.3 State Invariant

None

15.4.4 Assumptions

None

15.4.5 Access Routine Semantics

findPath($L, startId, stopId$):

- output: $out := \langle s : \text{String} \rangle$ that satisfies the following:
  - $out[0] = startId$.
  - $out[|out| - 1] = stopId$.
  - $out \subseteq \langle \forall (e : ParkingLayoutElemT | e \in L.getLayout() : e.getElemId) \rangle$.
  - $\forall (i : \mathbb{N} | i < |out| - 2 : L.getElem(out[i]).getType() = "Road")$.
  - $L.getElem(out[|out| - 1]).getType() = "Spot"$.
  - $\forall (i : \mathbb{N} | i < |out| - 1 : l.getElemIndex(out[i]) - l.getElemIndex(out[i + 1]) \in \{1, -1, l.getSize(), -l.getSize()\})$.
- exception: exc := No out can satisfy the requirements $\Rightarrow NoPathException$

16 MIS of Machine Learning Model Module

16.1 Module

ModelT

16.2 Uses

CameraCaptureT

16.3 Syntax

16.3.1 Exported Constants

None

16.3.2 Exported Types

ModelT = ?

16.3.3 Exported Access Programs

Name	In	Out	Exceptions
setModel	String		
setInput	String, String		
getResult		Seq of $\mathbb{B}$	

16.4 Semantics

16.4.1 State Variables

None

16.4.2 Environment Variables

model: the pre-trained machine learning model in the machine. The model will be deserialized as a python object.

16.4.3 State Invariant

None

16.4.4 Assumptions

None

16.4.5 Access Routine Semantics

setModel($pickleAdd$):

- transition: $modelAddress, model := pickleAdd, load(pickleAdd)$
- exception: $exc := \neg\exists(address : String|\neg address.exist()) \Rightarrow InvalidAddressException$

setInput($inputURL, inputTypes$):

- transition: $InputURL, Inputtype := inputURL, inputTypes$
- exception: none

getResult():

- output: $out := model.predict()$
- exception: none

17 MIS of Camera Capture Module

17.1 Module

CameraCaptureT

17.2 Uses

None

17.3 Syntax

17.3.1 Exported Constants

None

17.3.2 Exported Types

CameraCaptureT = ?

17.3.3 Exported Access Programs

Name	**In**	**Out**	**Exceptions**
getLatestFrame	String	Image File	
getLatestClip	String	Video File	

17.4 Semantics

17.4.1 State Variables

currentFrame: Image File
currentClip: Video File

17.4.2 Environment Variables

picamera

17.4.3 Assumptions

None

17.4.4 Access Routine Semantics

getLatestFrame(*directory*):

- transition: *currentFrame* := *picamera.capture(directory)*
- output: *out* := *currentFrame*
- exception: None

getLatestClip():

- transition: *currentClip* := *picamera.record(directory)*
- output: *out* := *currentClip*
- exception: None

18 MIS of User Action Handler module

18.1 Module

UserActionHandlerT

18.2 Uses

NavigationT, AuthT, AdminT, ModelT, ParkingStatsT, ViewT

18.3 Syntax

18.3.1 Exported Constants

none

18.3.2 Exported Types

UserHandlerT = ?

18.3.3 Exported Access Programs

Name	In	Out	Exceptions
handleChangeLayout	String, ParkingLayoutElemT		
handleParkingStats	ParkingLotLayoutT, Seq of $\mathbb{B}$	$\mathbb{N}$	
handleCheckAvailability		Seq of $\mathbb{B}$	
handleFindPath	ParkingLotLayoutT, String, String	Seq of String	
handleAuth	Sting, String	$\mathbb{B}$	

18.4 Semantics

18.4.1 State Variables

None

18.4.2 Environment Variables

None

18.4.3 State Invariant

None

18.4.4 Assumptions

None

18.4.5 Access Routine Semantics

handleCheckAvailability():

- output: $out := ModelT.getResult()$
- exception: none

handleChangeLayout($st, parkingele$):

- output: $out := AdminT.changeLayout(st, parkingele)$
- exception: none

handleParkingStats($layout, boolArray$):

- output: $out := ParkingStatsT.getStat(st, parkingele)$
- exception: none

handleFindPath($l, startID, stopId$):

- output: $out := navigationT.findPath(layout, String, String)$
- exception: none

handleAuth($id, pass$):

- output: $out := AuthT.authenticateUser(id, pass)$
- exception: none

19 MIS of View module

19.1 Module

ViewT

19.2 Uses

ParkingLotLayoutT, ParkingLayoutElemT

19.3 Syntax

19.3.1 Exported Constants

None

19.3.2 Exported Types

viewT = ?

19.3.3 Exported Access Programs

Name	In	Out	Exceptions
initLogin			
initPage			

19.4 Semantics

19.4.1 State Variables

None

19.4.2 Environment Variables

win: two dimensional sequence of coloured pixels

19.4.3 State Invariant

None

19.4.4 Assumptions

None

19.4.5 Access Routine Semantics

initLogin():

- transition: modify win with the following:
  - a text input field for a user name.
  - a text input field for a password.
  - a button to confirm the text inputs.
- exception: none

initPage():

- transition: modify win with the following:
  - use $showLayout()$ based on the location.
  - use $showLayout()$ to show a path.
  - a button to confirm logout.
  - a panel to show parking stats.
- exception: none

19.4.6 Local Functions

showLayout(L):

- transition: modify win so the elements $e : ParkingLotLayoutT$ of L are displayed on a $L.getSize()$ by $L.getSize()$ grid with the following table:

$getType()$	$getEnabled()$	$getHandicapped()$	$getReserved()$	$getOccupied()$	character
road					Black +
obstacle					Black O
spot	true	false	false	false	Green S
spot	true	false	false	true	Red S
spot	true	false	true	false	Green R
spot	true	false	true	true	Red R
spot	true	true	false	false	Green H
spot	true	true	false	true	Red H
spot	false				Grey S

showPath($L, startId, stopId$):

- transition: modify win first with $showLayout(L)$.
  $\forall(s : String | s \in NavigationT.findPath(L, startId, stopId)$:
  bolden the character on win corresponding to $L.getElem(s))$
- exception: none

References

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

20 Appendix

N/A