Name: Pouya Sameni

Student Number: 216491623

CSE Username: pouyas

Winter 2021

Table of Contents

Things to Note

When running the program please note that there are some pre-made accounts in the "DataBaseFiles" folder. Please use username: "admin" and password: "pass" to log into the admin account. Feel free to register a new account or use any of the premade accounts.

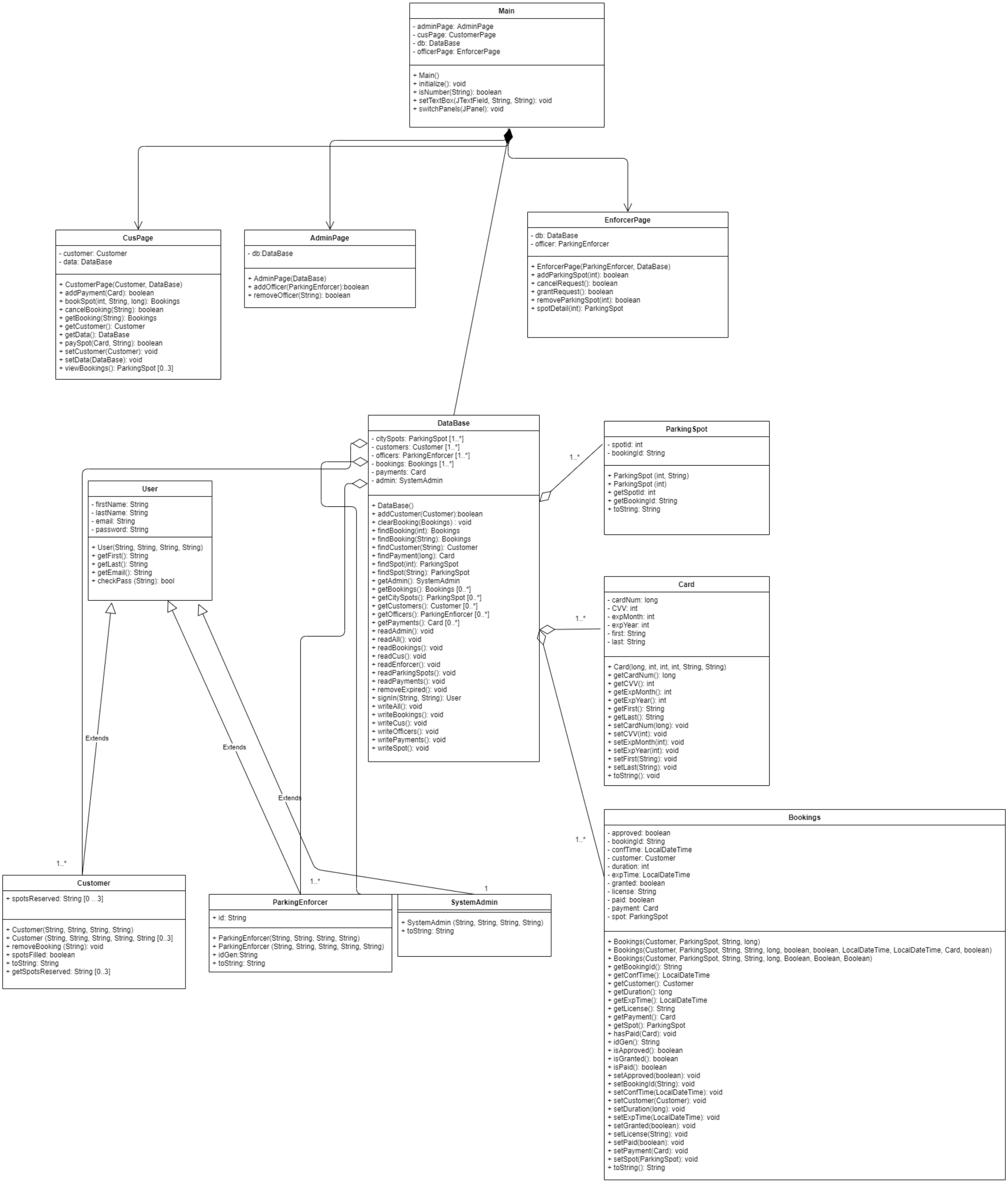
When booking a spot, an officer must first grant the book request and then customer will be allowed to pay for it. Once paid for admin can then view pending payments and confirm them. The system will automatically check if the name on payment is the same as customer booking.

Menus such as View bookings or any feature with a list can be used by selecting the desired object and clicking the "View Detail" button.

Many features were not able to be implemented using the corresponding "CustomerPage", "EnforcerPage" or "AdminPage" since it would make programming more complicated. Therefore many features were implemented in the Main class.

UML diagram does not include all attributes of the Main class such as the GUI attributes for text boxes and more. The only attributes included in UML are ones which are related to the other classes.

New UML Diagram



Main

- adminPage: AdminPage
- cusPage: CustomerPage
- db: DataBase
- officerPage: EnforcerPage

+ Main()
+ initialize(): void
+ isNumber(String): boolean
+ setTextBox(JTextField, String, String): void
+ switchPanels(JPanel): void

CusPage

- customer: Customer
- data: DataBase

+ CustomerPage(Customer, DataBase)
+ addPayment(Card): boolean
+ bookSpot(int, String, long): Bookings
+ cancelBooking(String): boolean
+ getBooking(String): Bookings
+ getCustomer(): Customer
+ getData(): DataBase
+ paySpot(Card, String): boolean
+ setCustomer(Customer): void
+ setData(DataBase): void
+ viewBookings(): ParkingSpot [0..3]

AdminPage

- db:DataBase

+ AdminPage(DataBase)
+ addOfficer(ParkingEnforcer):boolean
+ removeOfficer(String): boolean

EnforcerPage

- db: DataBase
- officer: ParkingEnforcer

+ EnforcerPage(ParkingEnforcer, DataBase)
+ addParkingSpot(int): boolean
+ cancelRequest(): boolean
+ grantRequest(): boolean
+ removeParkingSpot(int): boolean
+ spotDetail(int): ParkingSpot

DataBase

- citySpots: ParkingSpot [1..*]
- customers: Customer [1..*]
- officers: ParkingEnforcer [1..*]
- bookings: Bookings [1..*]
- payments: Card
- admin: SystemAdmin

+ DataBase()
+ addCustomer(Customer):boolean
+ clearBooking(Bookings) : void
+ findBooking(int): Bookings
+ findBooking(String): Bookings
+ findCustomer(String): Customer
+ findPayment(long): Card
+ findSpot(int): ParkingSpot
+ findSpot(String): ParkingSpot
+ getAdmin(): SystemAdmin
+ getBookings(): Bookings [0..*]
+ getCitySpots(): ParkingSpot [0..*]
+ getCustomers(): Customer [0..*]
+ getOfficers(): ParkingEnfiorcer [0..*]
+ getPayments(): Card [0..*]
+ readAdmin(): void
+ readAll(): void
+ readBookings(): void
+ readCus(): void
+ readEnforcer(): void
+ readParkingSpots(): void
+ readPayments(): void
+ removeExpired(): void
+ signIn(String, String): User
+ writeAll(): void
+ writeBookings(): void
+ writeCus(): void
+ writeOfficers(): void
+ writePayments(): void
+ writeSpot(): void

ParkingSpot

- spotId: int
- bookingId: String

+ ParkingSpot (int, String)
+ ParkingSpot (int)
+ getSpotId: int
+ getBookingId: String
+ toString: String

Card

- cardNum: long
- CVV: int
- expMonth: int
- expYear: int
- first: String
- last: String

+ Card(long, int, int, int, String, String)
+ getCardNum(): long
+ getCVV(): int
+ getExpMonth(): int
+ getExpYear(): int
+ getFirst(): String
+ getLast(): String
+ setCardNum(long): void
+ setCVV(int): void
+ setExpMonth(int): void
+ setExpYear(int): void
+ setFirst(String): void
+ setLast(String): void
+ toString(): void

User

- firstName: String
- lastName: String
- email: String
- password: String

+ User(String, String, String, String)
+ getFirst(): String
+ getLast(): String
+ getEmail(): String
+ checkPass (String): bool

Customer

+ spotsReserved: String [0 .. 3]

+ Customer(String, String, String, String)
+ Customer (String, String, String, String, String [0..3]
+ removeBooking (String): void
+ spotsFilled: boolean
+ toString: String
+ getSpotsReserved: String [0..3]

ParkingEnforcer

+ id: String

+ ParkingEnforcer(String, String, String, String)
+ ParkingEnforcer (String, String, String, String, String)
+ idGen:String
+ toString: String

SystemAdmin

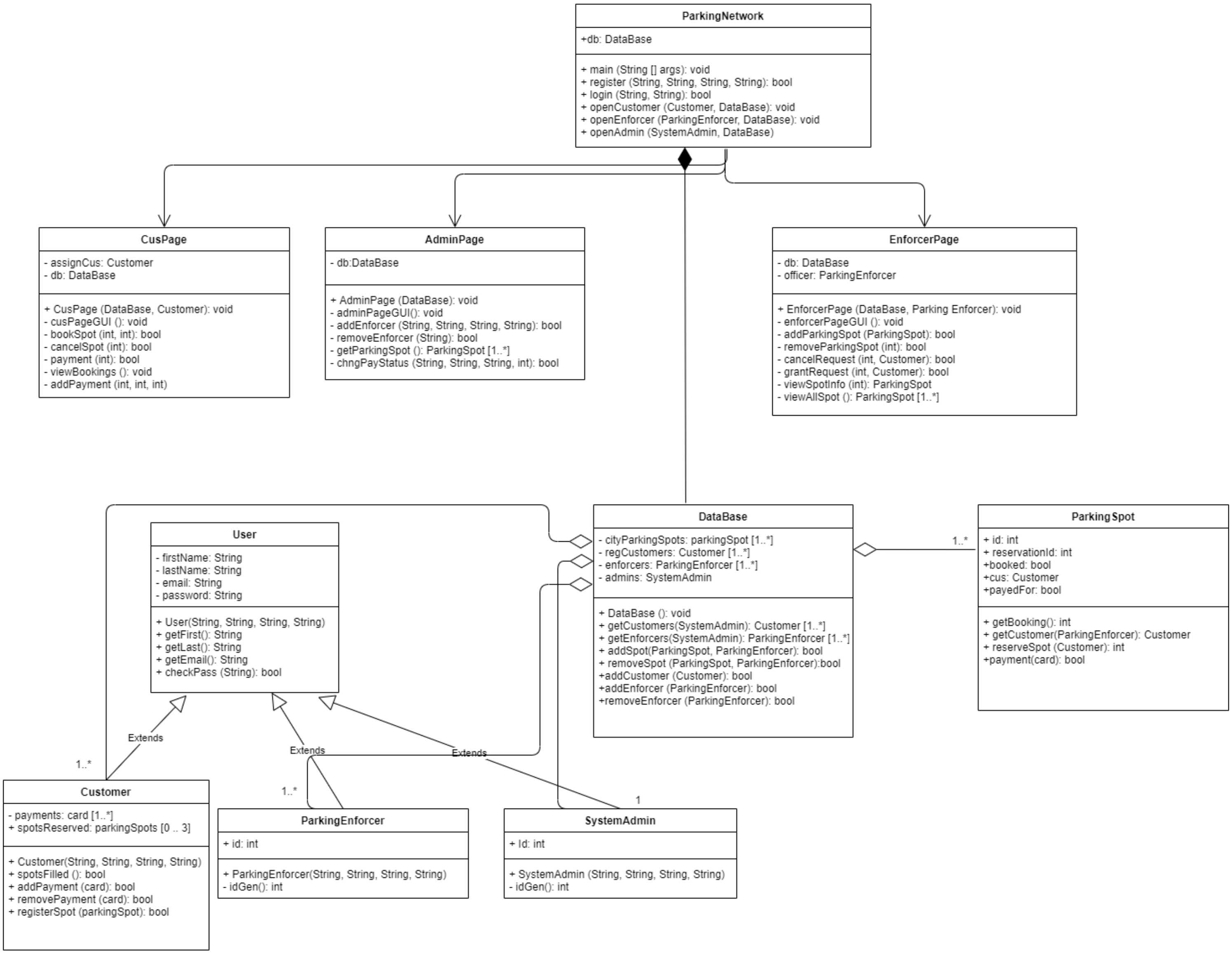
+ SystemAdmin (String, String, String, String)
+ toString: String

Bookings

- approved: boolean
- bookingId: String
- confTime: LocalDateTime
- customer: Customer
- duration: int
- expTime: LocalDateTime
- granted: boolean
- license: String
- paid: boolean
- payment: Card
- spot: ParkingSpot

+ Bookings(Customer, ParkingSpot, String, long)
+ Bookings(Customer, ParkingSpot, String, String, long, boolean, boolean, LocalDateTime, LocalDateTime, Card, boolean)
+ Bookings(Customer, ParkingSpot, String, String, long, Boolean, Boolean, Boolean)
+ getBookingId(): String
+ getConfTime(): LocalDateTime
+ getCustomer(): Customer
+ getDuration(): long
+ getExpTime(): LocalDateTime
+ getLicense(): String
+ getPayment(): Card
+ getSpot(): ParkingSpot
+ hasPaid(Card): void
+ idGen(): String
+ isApproved(): boolean
+ isGranted(): boolean
+ isPaid(): boolean
+ setApproved(boolean): void
+ setBookingId(String): void
+ setConfTime(LocalDateTime): void
+ setCustomer(Customer): void
+ setDuration(long): void
+ setExpTime(LocalDateTime): void
+ setGranted(boolean): void
+ setLicense(String): void
+ setPaid(boolean): void
+ setPayment(Card): void
+ setSpot(ParkingSpot): void
+ toString(): String

Old UML Diagram


ParkingNetwork
+db: DataBase
+ main (String [] args): void
+ register (String, String, String, String): bool
+ login (String, String): bool
+ openCustomer (Customer, DataBase): void
+ openEnforcer (ParkingEnforcer, DataBase): void
+ openAdmin (SystemAdmin, DataBase)
CusPage
- assignCus: Customer
- db: DataBase
+ CusPage (DataBase, Customer): void
- cusPageGUI (): void
- bookSpot (int, int): bool
- cancelSpot (int): bool
- payment (int): bool
- viewBookings (): void
- addPayment (int, int, int)
AdminPage
- db:DataBase
+ AdminPage (DataBase): void
- adminPageGUI(): void
- addEnforcer (String, String, String, String): bool
- removeEnforcer (String): bool
- getParkingSpot (): ParkingSpot [1..*]
- chngPayStatus (String, String, String, int): bool
EnforcerPage
- db: DataBase
- officer: ParkingEnforcer
+ EnforcerPage (DataBase, Parking Enforcer): void
- enforcerPageGUI (): void
- addParkingSpot (ParkingSpot): bool
- removeParkingSpot (int): bool
- cancelRequest (int, Customer): bool
- grantRequest (int, Customer): bool
- viewSpotInfo (int): ParkingSpot
- viewAllSpot (): ParkingSpot [1..*]
User
- firstName: String
- lastName: String
- email: String
- password: String
+ User(String, String, String, String)
+ getFirst(): String
+ getLast(): String
+ getEmail(): String
+ checkPass (String): bool
DataBase
- cityParkingSpots: parkingSpot [1..*]
- regCustomers: Customer [1..*]
- enforcers: ParkingEnforcer [1..*]
- admins: SystemAdmin
+ DataBase (): void
+ getCustomers(SystemAdmin): Customer [1..*]
+ getEnforcers(SystemAdmin): ParkingEnforcer [1..*]
+ addSpot(ParkingSpot, ParkingEnforcer): bool
+ removeSpot (ParkingSpot, ParkingEnforcer):bool
+addCustomer (Customer): bool
+addEnforcer (ParkingEnforcer): bool
+removeEnforcer (ParkingEnforcer): bool
1..*
Parking Spot
+ id: int
+ reservationId: int
+booked: bool
+cus: Customer
+payedFor: bool
+ getBooking(): int
+ getCustomer(ParkingEnforcer): Customer
+ reserveSpot (Customer): int
+payment(card): bool
Extends
Extends
Extends
1..*
1..*
1
Customer
- payments: card [1..*]
+ spotsReserved: parkingSpots [0 .. 3]
+ Customer(String, String, String, String)
+ spotsFilled (): bool
+ addPayment (card): bool
+ removePayment (card): bool
+ registerSpot (parkingSpot): bool
ParkingEnforcer
+ id: int
+ ParkingEnforcer(String, String, String, String)
- idGen(): int
SystemAdmin
+ Id: int
+ SystemAdmin (String, String, String, String)
- idGen(): int

Differences

When implementing the final design several factors had to be changed. Several classes had to be added and some had to be redesigned.

New classes included are Bookings and Card. Bookings is used to keep track of each booking made, it has all information such as payment, customer associated, and parking spot associated. In addition, it has expiry time, confirmation time and all Boolean fields used to track which stage of booking it is in. Card class is a simple implementation of a payment card with first and last name along with card info. This is used to load in payments and for admin to check payment against customer. Booking and Card are also written into a file like the other classes.

Data base implementation is roughly similar, various different helper methods have been added. Customer, Parking Spot and Bookings are all linked with the booking ID and only need to be loaded in once and then found using that identification. Although there is only one admin, a separate admin file is made for any future upgrades. Methods such as readAll() will read all files into the program upon start and will remove any expired bookings immediately. WriteAll() method is called after every major function and will remove expired bookings and write all object states to file. Object states are written to file using overwritten toString method.

GUI implementation is different than the midterm design. This is because it is much simpler to have one GUI class in main and perform all there in the same Jframe. This meant that some functions had to be implemented directly in main method without any helper class methods hence the simplicity of the page Classes. Page classes are just used to make implementation cleaner and organized. Any redundant checking is done in the specific class.

Overall the current UML and that designed for the midterm are very similar in design and implementation so I would say yes I did follow my original design closely.

Using the Program

Some features and text may not be exactly the same as the system specifications however they were implemented with what seems to make more sense considering my coding method and design. However final result does follow that of the system specifications.

Starting with running the program, all files are loaded in and user is greeted with start page, options are sign in and sign up. If signup then customer can make account and register, they then must click on sign in button to go to sign in page. If sign in unsuccessful then message is presented. Same sign in is used for admin and officers. Registration must have the @ in the email otherwise not valid.

Customer

Customer options are very straight forward. For booking, a valid spot must be entered all numbers, and a license plate. Dials can be used to select the amount of time for booking. Incrementing the minute does not loop around the hour as well so they are used independently. Minutes only increment 15 minutes to limit user from choosing obscure times. Maximum of 12 hours and 45 minutes can be chosen.

View bookings will only show bookings in list using booking id. If no bookings available, then message says so. In addition, price is calculated for $2 every 30 minutes.

Cancel booking must use valid booking id

Pay for booking will only show bookings that are granted by the officer after booking is placed. Second page of pay will have spots for payment info along with the info about the booking.

Parking Enforcer

Parking enforcer only has one option in main menu which is Manage Parking Spaces. In there a scrollable list of all parking lists will be presented. If a parking spot has a booking ID the list will present that. View details button will show more info and if not granted then the officer and grant or decline request. Add Spot and Remove Spot will lead to separate page where the officer can add or remove a vacant parking spot

System Admin

Admin add and remove officers are very straight forward however for managing payments. The list will only show bookings where a payment is made. Since the customer has already paid and system has checked if name on card is the same as booking the only option logically should be confirming the payment. This is because it would not make any business sense to deny the confirmation specially since the countdown has already been started and payment has gone through