REPORT 3

AUTOmatedPARK

Shravanthi Muthuraman, Phu Phan, Daniel Selmon, Madhumitha Harishankar, Joshua Beninson, Rashmi Loka

Effort Breakdown

Equal contribution.

Table of Contents

1. Summary of Changes

- Initially it was suggested that the users other than contract customers will be given the option to pick the spot in one of the given list of floors. But in order to avoid congestion at the entrance it was changed so that the system assigns the spot to the customer and displays it to him.
- Initially spot sensors were used only for the purpose of detecting vacancy but now it is also used to detect occupancy and make sure the user has parked in the assigned spot.
- Initially we thought we could implement random generation to generate inputs delivered by the hardware devices but we thought it would be more practical to give manual inputs during the demo to explain how the system works. When we have the actual hardware systems hooked up we will collect the inputs from them instead.
- The use cases were initially designed in such a way that there were many use cases which did not go well together to explain the flow of operation. This was because everything was broke up into very small pieces which could have been put together which would help improve the flow. This was implemented and the use cases were combined and modified to explain the flow of operation better.
- As the use cases were modified we also had to change the names of use cases to clearly represent the details described in each of the use cases.
- Even though combining the use cases made the use case look longer it helped us establish and deliver the flow much clearly which is much more important than making the use case shorter.
- The fully dressed description of use cases which were initially written for only some important use cases were modified and extended for all the use cases that were implemented. This was done because this would give the reader a step-by-step detailed explanation of the use cases and deliver the flow much clearly.
- Initially the sequence and interaction diagrams were drawn only for the use cases that were considered important but now sequence and interaction diagrams were drawn to represent all the implemented use cases. These were drawn in great detail and each alternate scenario was represented as a separate use case to avoid congested pictures.
  The new sequence and interaction diagrams are different from the older ones in that they are more detailed and based on our implementation.
- The domain model was revised to include the design changes that were made during the course of implementation. Moreover two additional views were included to explain the concepts in detail.
- Initially we thought we will need to maintain a database with just two tables for user details and reservations. But when we actually started the implementation of the database we ended up using more than just two tables. We had to create separate tables for login, member info, member address, reservation, and billing info.

- Before we had a traceability matrix only mapping the use cases to the functional requirements but now we extended the matrix to also include the mapping of use cases to nonfunctional requirements. This gives the reader a clear understanding of how the nonfunctional requirements relate to the use cases.
- The class diagram was revised to include and reflect the changes that were made during the implementation phase.
- The mathematical model which was initially designed to handle overstays and overbooking was no implemented but can be considered for future implementations.
- We initially thought of handling bill payment using a third party bill processing service such as PayPal to process payments but this idea was abandoned because we don't have actual payments to process and it would be a waste of money to buy a third party service. But when this has to be implemented in a actual garage payment processing can be done by just hooking up the third party service with the system and payments can be processed.
- The payment that registered customers owed was initially calculated on a regular timed interval basis but it has been redesigned in such a way that the amount will be calculated by a function when the user requests to view the bill or by the end of the month when the payment statement is being emailed to the user.
- The unit testing test cases were initially designed based on the system requirements but now it was modified to be based on the classes and use cases instead. Moreover unit testing was carried out in three different parts for the three units of the system such as classes, web pages, and database.
- The package diagrams were modified based on the feedback received and descriptions were added. The mapping of subsystems was done in a general context previously but now it is more specific and explains how each of the subsystem is connected.
- The persistent data storage is explained in more detail to include the specifications of data storage in the database
- Initially we were thinking of using network connections but later we were able to run the system from a single machine so our current implantation doesn't include any communication protocol.

2. Customer Statement of Requirements

Problem:

A multi-level parking garage is currently being operated without any computerized system. The current way of operating the garage is inefficient and causes customer dissatisfaction. The garage is not being operated to meet the goal of high occupancy to reap the most benefits. All the inefficiencies have resulted in a loss for the garage owner. The main problem that exists is the congestion inside the garage caused by drivers searching for a parking spot. In the case when a customer is unable to find a parking spot and decides to leave it is very unlikely that he would want to come back to the garage in future. This is because if only he knew that he was not going to get a parking spot he would have save himself at least 5 minutes of his time that he spent searching for a parking spot that he never found. This has a huge impact in the business since retaining customers is a major step to run a business successfully. There is no guarantee that a customer will have a parking spot when he arrives. In other words, there is nothing that a customer can do to reserve a parking spot. When leaving the garage customers have to pay the parking fare at the exit. This could result in congestion at the exit and delay the exit of customers. If this can be tackled effectively both customer satisfaction and garage occupancy can be increased. One other way to increase garage occupancy is by attracting more customers by using pricing strategies for which occupancy history is very important. Currently, the garage occupancy is managed by employees who walk around the garage to note down the occupancy of individual parking spots. This cannot give the owner or the administrator the best possible data to make decisions. We need more accurate data so that parking price levels can be determined according to the volume of occupancy. To solve all these problems and to make profit out of the garage business, the owner of the garage has decided to install a computerized system that would solve the above problems to the maximum extent possible.

Goal:

The main goal of the project is to design and implement a computerized system that will help maximize garage occupancy and increase customer satisfaction.

Description: Following is a detailed description of the functioning of the required computerized system and how it will be implemented.

Remodeling:

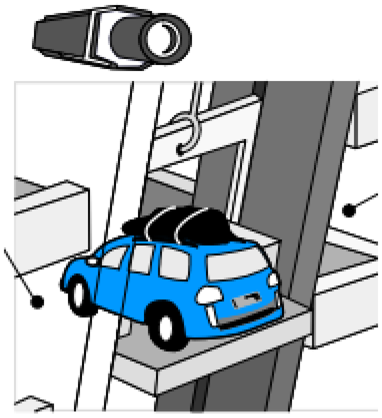
For the garage to be operated using a computerized system it has to undergo some remodeling. It will be remodeled in such a way that the only way to get to the upper decks of the garage is through a lift installed in the entrance of the garage. No cars can enter the garage without passing thorough the lift. The cars can exit the garage through the ramp to the ground floor. The garage is being remodeled to accommodate only passenger vehicles and cannot be use to park large vehicles such as trucks, busses, etc.

Devices:

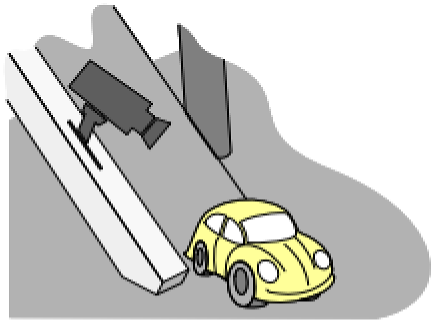
Instead of using electronic tags or markings for customer vehicle identification, the garage will make use of camera-based license-plate recognition systems. The following devices will be installed in the garage:

S1 & S2 (License plate readers):

S1.
Elevator camera



S2.
Exit camera



There will be two license plate readers installed in the garage. One will be on the lift platform and the other one at the end of the exit pathway in the ground floor. The plate readers will use a digital camera and a license-plate recognition system that are widely being used these days in toll stations in road-tolling systems. The plate readers read the license plate numbers upon the entry and exit of vehicles.

S3 (Spot Sensor):

S3. Occupancy photosensor

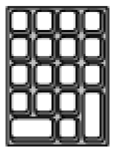
Spot sensors will be installed in all the parking spots. This will help us detect the availability and vacancy of parking spots. Once a car enters a spot the spot sensor will sense it and marks the spot as unavailable in the database. Once a car leaves the spot, the sensor will sense it and the spot will be made available again in the database.

D1 (Display Monitor):

Park-a-lot
Your spot #:
3014

There will be a display monitor installed in the elevator. This will be use to communicate with the user once he enters the lift. This is mainly used to display the assigned spot to customers. This will help us handle situations when the number plate is not recognized by the plate reader and to assist walk-in customers.

D2 (Keyboard):



There will be a key board attached to the display monitor to help the user interact with the system. The user inputs the data requested by the system (login details, walk-in agreements, etc) using the key board.

There will be two one-way barriers in the garage, one in the entrance pathway to the ground floor and one at the exit. This will prevent drivers from entering the garage unauthorized.

Types of users:

1. Registered users: Users who have registered with AUTOmatedPARK through the website.
    a. Contract users: Users who have made a monthly contract with AUTOmatedPARK and will have a guaranteed parking spot for the registered month.
    b. Reserved users: Users who have made a prior reservation for a particular duration of time.
    c. Walk-in users: Users who have an account with AUTOmatedPARK but did not make a reservation or a contract for by the time they want to use the garage.
2. Unregistered users: Users who have not registered with AUTOmatedPARK but would like to use the garage. They will be treated just like registered walk-in users.

Note: Anyone who wishes to park in the garage but doesn't have a reservation or contract will be considered a walk-in user.

Working of the system:

Users who want to reserve a parking spot in the garage will have to register with AUTOmatedPARK. Their username will be their email id and they can pick a password at the time of registration. Users can login to their user account to make reservations, manage reservation, change user information etc. Users can make monthly contracts or reservations. Reservations have to be for a minimum of 30 minutes. When the user wants to create an account the system asks the user for email id, password, first name, last name, mailing address, credit card details, and cell phone number. The above fields are required fields and one of the optional fields would be a default number plate number. The default number plate number can be useful when the user want to make a reservation and doesn't want to enter the plate number every time. Once the user agrees to the terms and conditions the system will create a row for the user in the database and maintain the account. The account will be deleted after five years of inactivity. This is done so that the system is maintainable over a long period.

When a user wants to make a contract and selects a month the system checks to see if there are available spots for that time duration. If yes then it updates the database in such a way that a spot is always reserved for that user in a particular floor. The number of available spots and unreserved spots in a particular floor are decreased by one and are not incremented until the end of the month. This is done because we don't want to lose contract customers. When a user wants to make a reservation and selects the date and time then the system check for the availability of a spot. If there are spots available then the system reduces the number of unreserved spots by one for that time interval.

When the user arrives at the parking lot the system gets data from the license plate reader installed in the lift to determine if the user has a reservation or contract. If he has a contract then the assigned spot is displayed to the customer and the car is taken to the floor. The database will be notified of the customer's arrival. This will be used to maintain customer history. If the user is a reserved customer the system would check the database for an available spot in one of the upper floors. Once the system finds a spot it assigns the spot to the customer and displays it to the customer. The number of available spots in that floor will be decreased by one. The car will be lifted to the corresponding floor. Once the car occupies the assigned spot, the spot sensor senses it and updates the spot as occupied in the database.

If a reserved user arrives at the garage during the reserved period of time and there are no available spots in the upper levels, the system will check to see if there are any spots available in the ground floor. If there are spots available in the ground floor he will be allowed to park in the ground floor for the same rate that he will be charged for the reservation. If there are no spots available even in the ground floor then the user will be asked to leave and to compensate for the inconvenience he will be issued a rain check.

An unregistered customer or a registered customer who has not made any contract/reservation but would like to use the garage will be considered as walk-ins who will be asked to park in the ground floor if there is any spot in the ground level and charge them at a higher rate. If there are no available spots in the ground floor the user will be asked to leave. If the user is a registered customer then it will match the number plate number with the user account for the current period of stay (until he exits the garage) and update the database. If the user is unregistered the system creates a temporary account and feeds in the plate number. This account will be maintained until the period of stay.

Once a car leaves a spot the spot sensor senses it and the system updates the database. If it is not a contract customer the number of available spots in the floor is increased by one. Users in the upper deck can exit the garage through the exit ramp. Once the license plate reader recognizes the number then it opens the gate and the user can exit. Registered users in the ground level can exit once the camera recognizes the plate number and opens the gate. In both cases the user accounts will be updated in the database. Unregistered customers pay the parking fare before they exit. The operator will be notified and he will help the user pay their bill. After this the gate will open and they can exit the garage. Their account will be deleted from the database once they pay the bill.

If we are unable to server customers who have arrived at the garage due to various reasons such as unavailability of spots (overstays, overbookings, etc) or at the customer’s will then they will have to leave the garage. Backing out might cause problems if there are cars waiting in line. So there will be a side path that the users can use to leave the garage.

Business Policy:

P1: If the license plate number is not recognized by the system, the user won’t just be asked to leave. The user will interact with the customer through the display monitor and key board to determine the best possible solution. The plate number might not be recognized if the user uses a rented car or a car whose plate number does not match the number in the system. In this case the user will be asked what type of user he is and provides service accordingly.

P2: If a reserved/contract user’s plate number is not recognized then the user will be asked to enter his login details. Once the login is verified and the system finds the required fields the user will be able to use the garage. The system matches this new plate number to the user’s account for the current period of stay (i.e. until the user exits). The assigned spot will be displayed to the customer and he will be transported to the appropriate floor.

P3: If a registered user who has not made or contract/reservation is not recognized the user will be asked for his login details. Once the login is verified and the system finds the required fields, it will check to see if there is an available spot in the ground floor. If so the user will be asked if he would like to use the garage as a walk-in customer for the specified rates. If the user agrees then the system matches this plate number to the user's account for the period of stay and lets the customer into the ground floor by opening the barrier gate.

P4: Contract user may arrive any time during which he has a reservation. A reserved customer must arrive within a certain grace period to be able to use the reserved spot. If a reserved customer arrives during the grace period to the parking lot he will have the option to extend his reservation time if there are available spots. If not the customer will just be allowed to use parking for the reserved period of time and after that it'll be considered as overstay. He will have the option to leave the garage if he doesn't agree to any of these terms.

P5: If the user doesn't arrive within the reserved period of time he will be charged for the entire reservation time.

P6: If a user decides to leave the garage before the reserved period of time, he will still be charged for the entire reservation period.

P7: Users who have made a reservation can extend their reservation anytime before the end of the reserved duration provided there are available spots.

P7: If a user wants to cancel a particular reservation, it has to be done at least 30 minutes (decided by administrator) prior to the end of the reservation period. If it is canceled successfully then the user will not be charged anything. If not then the user will be just charged for the entire reservation period (P5).

P8: Walk-in customers (unreserved registered and unregistered customers) will be charged at a higher rate than reserved/contract customers. The price ranges will be decided by the administrator.

P9: Customers who do not leave the garage before the end of the reservation period will be charged at overstay rates for the time of stay after the end of the reservation period. Overstay customers will receive a text message warning them of the overstay policy.

P10: Contract customers have a guaranteed spot.

P11: If a reserved customer arrives at the garage and there are no available spots in the upper decks due to overstays then the system checks to see if there are available spots in the ground floor. If there is an available spot then the customer will be allowed to use the ground floor and pay the regular fare as a reserved customer. He won't be considered a walk-in and billed at a higher rate.

P12: If a reserved customer arrives and the garage is full (not available spots in any of the floors), the user will be asked to leave the garage and will not be charged for the reserved duration. To compensate for the inconvenience he will be issued a rain check.

P13: A user can have multiple standing reservations under his account but these reservations cannot be contiguous. A minimum gap will be imposed between consecutive reservations. If it is less than that minimum gap the user will be asked to merge the reservation. A user can have up to 3 outstanding reservations.

P14: More than one user cannot use the same number plate to make reservations or use the garage for the same time duration.

P15: Registered users have to log into their account in order to pay their monthly bills. They can either use a credit or a debit card to pay their bills. In our efforts to go green we will not be mailing out monthly bills. Instead an email reminding users of their monthly bills will be sent.

P16: Registered users have until the 10th of every month to pay their last month bills. Failure to do so will result in an addition of late fee to their monthly bill. Users can pay their monthly bill with the late fee until the 20th of next month. If we do not receive a payment from the user by 20th an automatic deduction will be made from the user's account which was provided during registration.

P17: Unregistered walk-in customers will have to pay the parking fares in order to be able to exit the garage.

P18: If a user wants to make a reservation it has to be for at least 30 minutes.

Assumptions about the system-to-be:

A1: The license plate readers will work correctly 100% of the time with no errors. The sources of error could be dirty or scratched number plates, missing number plate, etc. But we assume that these errors do not occur and prevent the license plate reader from reading the correct plate number.

A2: In the cases when the system cannot accommodate the user or the user does not agree to the terms and conditions of the system, the user will be asked to leave. We assume that the user will always obey and leave.

A3: The lift will work perfectly at all time. It will lift the car to the correct floor and will not stop anywhere in between or in the wrong floor.

A4: A customer parks only in one parking spot and does not park in such a way that one car occupies two spots.

A5: In the case of assigning spots we assume that the user just accepts the spot and doesn't demand for a different spot.

A6: The spot sensors work perfectly 100% of the time and do not malfunction. The spot sensors will recognize the spot as occupied only if there is a car in the spot. For example a sensor based on visible light will detect use the darkness of around it to detect the occupancy of the spot. In such a case we assume that the darkness would be caused only due to occupancy of a vehicle.

A7: The user will proceed to exit immediately after he leaves the parking spot.

A8: It is possible that the customer is an organization with multiple individuals (rather than an individual person). This is possible because we are identifying customers based on the number plate and not the individuals.

A9: The user has an email account and a cell phone with texting enabled. This is important because email id is used as the username for login. Texting is used to warn users of overstays.

Overbooking:

In order to make the most out of the business we will also implement overbooking which is done in most of the leading business such as airlines and hotels. We will use a definite formula to determine the number of overbookings we can process without losing customers. The formula used will be as follows:

  total number of parking spots available

- (confirmed reservations) x (no-show factor based on historical data)

- (guaranteed reservations) x (no-show factor based on historical data)

- predicted overstays

+ predicted under stays

- predicted walk-ins

= number of additional parking spots available to achieve 100 % occupancy

3. Glossary of Terms

- Administrator is the system manager and he/she will have elevated rights to change prices and view garage history.
- ADO.NET (ActiveX Data Objects for .NET) is a set of computer software components that programmers can use to access data and data services.
- Android is a Linux-based operating system for mobile devices such as smartphones and tablet computers. It is developed by the Open Handset Alliance, led by Google, and other companies.
- ASP.NET is web application framework developed and marketed by Microsoft to all programmer to build dynamic Web sites, web Application and web services.
- Compensation is the credit that the customer receives if they are not able to park on their reserved spot because of overbooking or other customers who failed to depart as scheduled.
- Confirmed reservations are present registered customers who make a reservation. These reservations are honored until a specific time (including the grace period after the start of the reserved interval). Such customers represent the critical element in no-shows.
- Customer is the actual person that utilizes the parking garage by reserving or walking in.
- Database is an organized collection of data for one or more purposes, usually in digital form.
- Display monitor is a television screen installed in all the parking spots. This will help us detect the availability and vacancy of parking spots and display other messages.
- End user Please refer to customer.
- Grace period is the time period that the parking spot will be reserved after the start of the reserved interval. If the customer arrives within the holding period, he or she shall park on their reserved spot and will be billed for the full reserved period. The customer will be offered to pay an additional fee to hold the reservation beyond the regular grace period
- Ground floor: The lowest floor of the garage. This floor is used mostly by walk-in customers.
- Guaranteed reservations represent the registered customers who made contact with the parking garage for a parking spot, such as commuters going to work who need parking on a daily basis during a monthly period. Such customers represent a less volatile group because of the need to show up for their work.
- GUI (Graphical User Interface) a type of user interface that allows users to interact with electronic devices with images rather than text commands. A GUI offers graphical icons, and visual indicators, typed command labels or text navigation to fully represent the information and actions available to a user.
- HyperText Markup Language (HTML) is the main markup language for web pages. HTML elements are the basic building-blocks of webpages.
- Institute of Electrical and Electronics Engineers (IEEE, read I-Triple-E) is a non-profit professional association that is dedicated to advancing technological innovation and excellence.
- Internet Protocol (IP) is the principal communications protocol used for relaying datagrams (also known as network packets) across an internetwork using the Internet Protocol Suite
- License-plate readers are the devices used to recognize the license plate number of the car. These devices are put on platform and at the end of the exit pathway.

- Mobile interface is a minimalistic interface made for customers accessing the website via mobile phone.
- Monthly contract is an agreement between the customer and the parking system to agree to pay the monthly fee and have a spot reserved for him/her for the entirety of the month.
- Monthly statement is the email that the registered customer receives once a month, including parking fee or penalty.
- No-show is when the customer does not show up for their reservation.
- Overbooking is when the number of reservations exceeds the capacity of the parking garage.
- Overstays are customers who arrive on time but decide to leave after their predicted time of departure.
- Owner is the owner of the parking garage.
- Password is a string of characters that is used for authentication.
- Registered customers are customers having accounts with the system. The system has data of the customers.
- Server is a computer program running to serve the requests of other programs. It executes the information between the user, database and the data sent from the garage.
- Software application architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes such as performance, security, and manageability. It involves a series of decisions based on a wide range of factors, and each of these decisions can have considerable impact on the quality, performance, maintainability, and overall success of the application.
- Spot sensor Devices installed in all the parking spots. This will help us detect the occupancy and vacancy conditions of parking spots.
- Unregistered users Users who does not register for an account with the system. These customers will be treated as registered walk-in users.
- Walk-in customers are registered customers who arrive to the parking without a contact or reservation. They are well come because they can enhance the garage occupancy percentages (if there are available parking spaces).

4. System Requirements

A. Enumerated Functional requirements

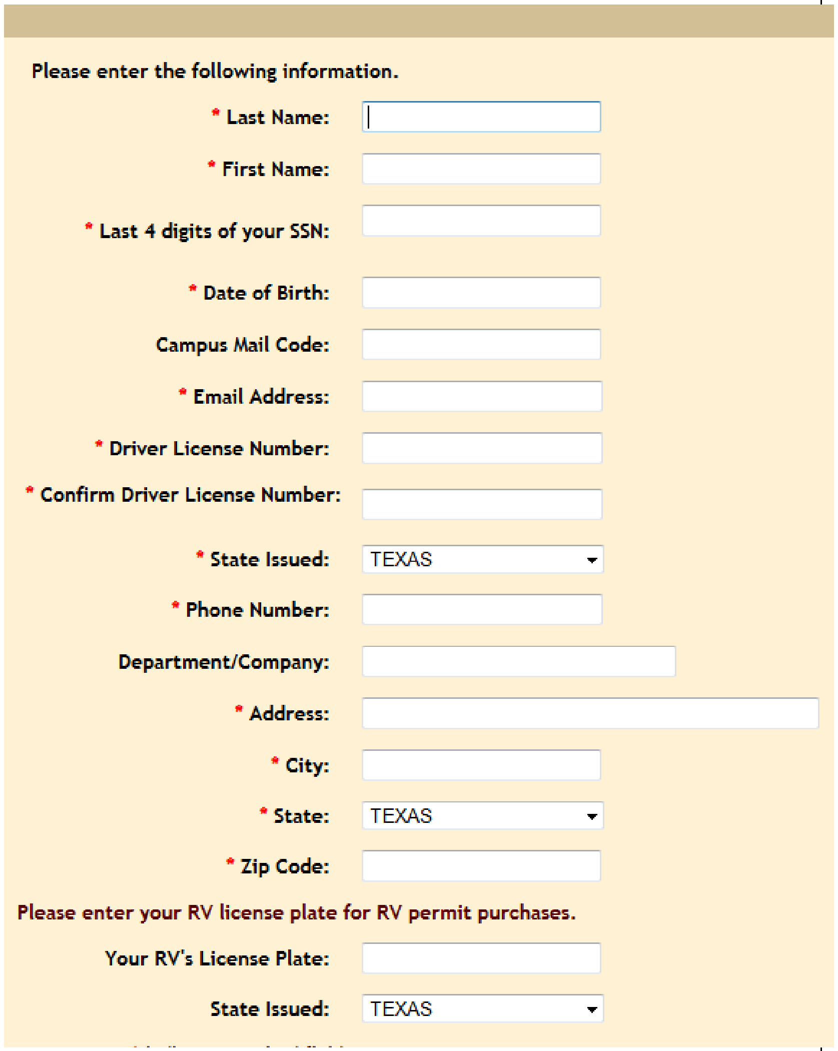
REQ No.	Priority Weight	Description
REQ-1	2	New customers will have to be able to register with AUTOmatedPARK so that they can make future reservations.
REQ-2	1	If a registered customer wants to make a monthly contract he will be given a list of months to choose from. Based on the month he selects and the availability of spots the database will reserve a guaranteed spot for the user.
REQ-3	1	When a registered customer wants to make a reservation for a particular time period then he'll be given a list of dates and times to pick from.
REQ-4	2	The administrator should be given special privileges such as access to user history, occupancy records, and authority to set price ranges.
REQ-5	5	Once the presence of a car in the lift is detected, the license plate reader has to read the plate number and check to see if a contract/reservation has been made with that number.
REQ-6	4	If the plate number of a car is not recognized by the license plate reader as reserved/contract the system will interact with the user to determine how the user can be offered service.
REQ-7	4	If a reserved/contract customer's plate number is not recognized the user will be asked to input his log in details and this will be used to access the user's account and provide service accordingly
REQ-8	4	If a registered customer's plate number is not recognized the user login details will be requested and the customer will be considered a walk-in customer
REQ-9	4	If an unregistered customer arrives to park in the garage he will be treated as a walk-in customer.
REQ-10	2	Once the system determines which floor to transport the car to, the elevator has to lift the car to that deck.
REQ-11	4	Once a user leaves a spot the spot should become available immediately and not be unavailable till the car exits the garage. This would reduce the chance of turning down customers due to unavailability of spots.
REQ-12	4	If an unregistered customer wants to leave the garage the operator (security guard) has to be notified. The operator has to help the customer pay his bill in order to be able to exit the garage.

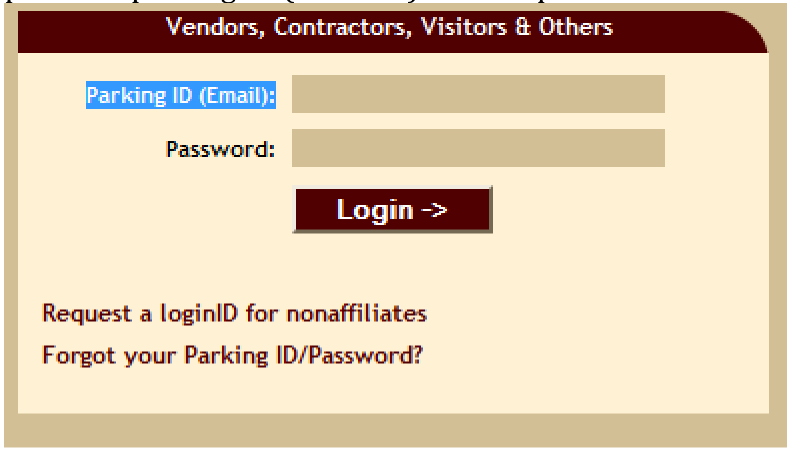
REQ-13	3	If the user cancels a reservation 30 minutes prior to the start of the reserved time period, the reservation has to be cancelled and the database should be updated. The user should not be charged any price for the cancelled reservation.
REQ-14	3	If a user extends the duration of reservation before the end of the reserved time period, the database should be checked to see the availability of spots. If there are available spots the reservation should be extended and the database should be updated. The user should not be charged overstay fees for the extended duration of time.
REQ-15	3	If a reserved customer doesn't arrive within the grace period the reservation has to be cancelled but the user will be charged for the full reserved period.
REQ-16	2	The exit gate has to open for registered customer once the license plate reader at the exit reads the plate number.
REQ-17	3	The gate that leads to the ground floor should open to permit walk-in customers to enter the ground floor. This should be done after an available spot has been detected and the user agrees to the walk-in terms and conditions.
REQ-18	4	If the upper decks of the garage are unable to accommodate a reserved customer (due to overstays and over booking) he has to be able to park in the ground floor if there are spots available in the ground floor.
REQ-19	3	If it is impossible to accommodate a reserved customer in any levels of the garage the customer will be asked to leave the garage and will not be charged for that particular reservation. To compensate for this the user will be issued a rain cheque.
REQ-20	1	The user has to be able to manage his account details by logging into his account. For instance the user has to be able to change the number plate that he wants to use as a default one, change password, address, etc.
REQ-21	2	If a reserved customer stays in the garage for longer than the reserved period of time he should be notified of the overstay policy and be charged at a higher rate for the overstay period.

B. Enumerated Nonfunctional Requirements

#	Priority Weight	Description
REQ-22	5	AUTOmatedPARK should only permit authorized users to access/edit a customer's profile. Customers should not be able to alter their amount owed in any way other than posting a payment.
REQ-23	5	Only a valid email address and credit card number should be accepted.
REQ-24	4	Only those customers with an account profile in good standing should be allowed to make a reservation.
REQ-25	3	AUTOmatedPARK should offer several modes of account access options (i.e. mobile app, garage consoles, and computer access) to facilitate ease of use. This pertains to being able to view/interact with reservations, money owed and account standing so as to facilitate the highest degree of customer service.
REQ-26	3	All interfaces should be simple yet inclusive enough to allow any use case to be executed in at most 5 minutes.
REQ-27	4	An accurate tally of spot availability, at any point in time, for each customer type should be kept as accurate as possible. The spot sensors should provide immediate details in proper usage conditions. However, sensor failure is to be expected, so it is necessary to provide at least 2 sensors in different locations on the floor of each spot as a failsafe. Any contradicting input from the sensors will trigger a notification to the operator to inspect the spot for debris or failure. This should bring average spot accuracy above 95%.
REQ-28	5	An accurate database should always be maintained for each customer's profile which includes: usage history, amount due (if delinquent, the length of such), past payment history, customer type, upcoming reservations (if applicable), contract period (if applicable), associated license plates, and membership status.
REQ-29	3	AUTOmatedPARK should be able to admit a customer to the ground floor from the entry console within 1 minute time.
REQ-30	3	A customer utilizing the elevator should have access to their floor within 3 minutes from the entry console.
REQ-31	4	AUTOmatedPARK should incorporate self-check features to get malfunctioning hardware serviced/replaced to minimize any reduction of service

C. On-Screen Appearance Requirements

REQ No.	Priority weight	Description
REQ-32	5	In order to make an account online, the user needs to have Internet-enabled devices and access the home page of the parking garage system. The user should be able to input registration information such as: last name, first name, date of birth, email address, phone number, etc.  Please enter the following information. * Last Name: * First Name: * Last 4 digits of your SSN: * Date of Birth: Campus Mail Code: * Email Address: * Driver License Number: * Confirm Driver License Number: * State Issued: TEXAS * Phone Number: Department/Company: * Address: * City: * State: TEXAS * Zip Code: Please enter your RV license plate for RV permit purchases. Your RV's License Plate: State Issued: TEXAS source: https://transport.tamu.edu/account/NARequest/NAAccount.aspx

REQ-33	5	To login into the system, the customer should be able to put their parking ID (or email) and the password.  Vendors, Contractors, Visitors & Others Parking ID (Email): Password: Login -> Request a loginID for nonaffiliates Forgot your Parking ID/Password? source: https://transport.tamu.edu/account/
REQ-34	5	To make a reservation online, the user would need to log on their account on the internet page or the app on their phone. Users would also need a page for reserving and input the information for the reservation: time (day, month, year), location (because the system has several parking garages located in different place), number plate, etc.
REQ-35	4	If users want to cancel, modify, extend their reservation, they need to log into their account and choose whether they want to cancel or just modifying/extending; if they want to modify/extend, they need to add the additional information: time, day, location, parking period, the number plate of their car, etc.
REQ-36	4	In order to edit the information of their parking account, the user needs log on the webpage of the garage, click on an edit account button/link. After that there should be several fields of information, the user should be able edit any field they want (including address, city, state, zip code, credit card number, new password, old password, etc.) then click on submit. The user should not be able to edit the e-mail, because it is used to identify the user.

5. Functional Requirements Specification

A. Stakeholders:

1) Customer - Owner of the garage
2) Construction architect
3) Maintenance officials
4) Security guard/operator
5) Database architect
6) Web-page designers
7) Developers
8) Users - Registered, Unregistered, Reserved, Contract, Walk-in
9) Project Manager
10) Business Analyst
11) Third party bill payment services

Some of the stakeholders such as third party bill payment services are not involved in the current implementation of the project but will be of interest when the system is modified to include some of the suggested developments.

B. Actors and goals:

Following is the list of actors (people or devices) that will directly interact with the system, their types (initiating or participating), and their goals.

<u>Initiating</u>
Owner:

- To set prices and rates.
- To view history and occupancy of the garage.

User:

- To register, manage account and pay monthly bill on the website.
- To make a reservation for a particular time period or sign a contract for long term reservations (eg: monthly reservations).
- To enter registration details if the license plate number is not recognized.
- If unregistered, to pay the parking fare upon exit.

Participating

License plate reader 1:

- To read the license plate number of a vehicle once it enters the lift upon entrance.

License plate reader 2:

- To read the license plate number of the vehicle as it exits the garage.

Lift monitor:

- To enable interaction between the users and the system by displaying details such as spot availability and terms.
- To direct the user to the assigned spot if he is assigned one.

Keypad:

- To help users make selections and input registration and reservation details as necessary.

Elevator:

- To lift a vehicle to the corresponding floor of the assigned spot.
- To provide an entrance for vehicles to access the garage.

Payment terminal:

- To process payments of unregistered customers at exit.

Spot sensors:

- To realize states of occupancy or vacancy of parking spots.

Operator:

- To help unregistered customers pay his/her parking fare upon exit.
- To attend to any customer issues at the garage.
- To notify the maintenance officials about any device malfunctions or problems.

Maintenance manager:

- To inspect the garage and devices regularly to ensure customer satisfaction.

Gate/Barrier:

- To allow for entry into the lower lever and exit from parking lot.

Webpage:

- To allow customers register with the system and make reservations.
- To help users manage their account

Database:

- To maintain a record of parking spots and their availability.
- To manage registrations and reservations.
- To maintain history of customer overstays and usage of the parking lot.

C. Use cases:

The following is the list of the use cases for the system and their description.

i. Casual Description:

UC-1 Register:

In order to register with the system the user has to go to the web site and create a user account. The user will be asked for some basic necessary details such as first name, last name, mailing address, phone number, credit/debit card number, e-mail, password, and security question. One optional field would be a default license plate number. Once the user has entered all the required details the policies page and the rates set by the owner of the garage will be displayed. When the user accepts the terms and conditions, an account will be established and maintained. The user can login with the e-mail and password at any point to make or cancel reservations and make changes to the account. If the account is continuously inactive for more than one year the account will be deleted from the database.

FUTURE WORK CONSIDERATION IS TO IMPLEMENT UC-1 WITH EMAIL AND CREDIT CARD VERIFICATION: Non-Functional Requirement #23

UC-2 Reserve:

Once the user has registered he/she can use the username and password to login to make reservations. Unregistered customers cannot make reservations. When the user clicks the make reservations button he will be asked if he wishes to make a monthly contract or just register for a particular period of time. Upon selection of contracts the user will be given a list of months to pick from. Upon selection of regular reservation the user can pick the time slot that he/she wants to reserve the spot. The program will check if there is an available spot for that selected time period and if there is a spot available the user will be notified of the total amount for the reservation based on the rates set by the administrator. Then the user will be prompted to enter an optional license plate number. He will be given the option to use the default number used during registration. Once the user enters an optional license plate number and agrees to the terms a spot will be reserved for the user and the database will be updated.

Alternate Scenario:

If there are no spots available during the time period requested by the user then the will be advised to check back later because spots can become available due to possible cancellation of reservations.

FUTURE WORK CONSIDERATION IN UC-2 IS TO VERIFY USER HAS AN ACCOUNT IN GOOD STANDING: Non-Functional Requirement #24

UC-3 EnterLift:

Once a car enters the lift, license plate reader 1 is activated and it reads the license plate number of the car. If the number is recognized as belonging to a reserved user, the software checks to find the floor with a vacant spot. Once it finds one, the elevator console will display the assigned spot where the user can park. The customer will then be transported to the appropriate floor. In the meanwhile, the database will decrement the number of free spots available on that floor. Once a car occupies a parking spot the spot sensor would detect occupancy and mark the spot as occupied in the database.

Alternate Scenario1:

If there are no spots available in the upper levels, the system checks to see if there are spots available in the ground floor. If there are spots available in the ground floor, a spot is assigned to the user and he is let into the ground floor where he can park. If there are no spots available in the ground floor, the user will be asked to leave and will be issued a rain check.

Alternate Scenario 2:

If the number is recognized as belonging to a contract customer, the customer is notified the spot number that he needs to park in. The customer will be taken to the appropriate floor. The database will not be alerted of a decrement in free spot count. Contract customers have a spot marked occupied for them regardless of presence.

Alternate Scenario 3:

If the license plate reader reads a number that is not in the database of reserved or contract customers the display monitor allows the customer to choose between options ‘registered’ or ‘reserved’. Based on the sequence of selections the type of customer is determined. Depending on the type of customer the customer is serviced as follows:

Registered Reserved customer:

If the customer is registered, he is asked for the login details. If the software detects that the user has made a reservation the software will link the current plate number to the reservation. Then it will follow the same procedure that would be used to service a reserved customer.

Registered Unreserved customer:

If the customer is registered, he is asked for the login details. If the software detects no reservation made for the user, it checks to see if there is a vacant spot in the ground floor. If available, the user will be asked if he accepts to the terms and conditions and be a walk-in customer. If the user accepts the software links the plate number to the user account. The console displays to the customer the spot where he can park and opens the gate that leads to the ground floor. The user can then park his car in the assigned spot. The database will be updated to decrement the number of available spots in the ground floor and the spot sensor updates the spot as occupied. If there are no spots available in the ground floor, the user will be asked to leave.

Unregistered customer:

If the customer is unregistered, the software checks to see if there is a vacant spot in the ground floor. If there is a spot, the user will be asked if he accepts to the terms and conditions and be a walk-in customer. If the user accepts the software creates a temporary account and maps the plate number to this temporary account. The temporary account will be maintained for the customer’s duration of stay in the garage. This is done for billing purposes. The console displays to the customer the spot where he can park and opens the gate that leads to the ground floor. The user can then park in his car in the assigned spot. The database will be updated to decrement the number of available spots in the ground floor and the spot sensor updates the spot as occupied. If there are no spots available in the ground floor, the user will be asked to leave.

UC-4 Overstay:

If a reserved or contract customer does not leave the parking spot within the reserved period of time he will be charged at a higher rate for the extra time that he uses the parking garage. The database will be checked every 30minutes (this interval depends on what time basis the administrator wants to charge the customer) to check for overstays. After the first check during which a car is found to be overstaying the system will send out an email warning the user of the overstay policy.

UC-5 UserManage:

If a user wants to cancel or extend any previously made reservations he can do so by clicking the "Manage Reservations" option under the Manage tab. Once the user goes to the manage reservations page and clicks cancel reservation, the program will check if it is at least 30 minutes prior to the start of the reservation period. If it is, then the user will be allowed to cancel his reservation. If not then the user will not be able to cancel the reservation and will be charged for the reserved period of time.

FUTURE WORK CONSIDERATION IN UC-5 IS TO OFFER A MOBILE APP WITH WHICH TO ACCESS THEIR ACCOUNT: Non-Functional Requirement #25

Alternate Scenario 1:

If a user wants to extend a reservation the program will check to see if it is before the end of the reservation period. If so then it'll check to see if there are any available slots. If there is availability the user will be allowed to extend the reservation time for a specified time period and the database will be updated. If there are no available spots then the user will not be able to extend the reservation period.

Alternate Scenario 2:

If a user wants to edit the details provided during registration he can do so by selecting "Manage Profile" option under the Manage tab. The user will have access to a to manage his profile by changing initially input registration details such as address, phone number, credit card information, license plate number, etc.

UC-6 Exit:

Once a car leaves, the spot sensor marks the spot as vacant again. The database will be updated once the sensor detects the vacancy. If it is not a contract customer then the number of available spots in that particular floor will then be increased by one.

Upon leaving the spot, the most likely thing a customer will do is exit the garage. As the customer leaves the garage the license plate reader 2 near the exit of the garage will detect the number and send the information to the program to check the customer out of the garage.

UC-7 SetPrice:

When the administrator logs on with administrative privileges, he/she will be allowed to set or adjust prices for reserved, unreserved and contract customers. To help the administrator determine the correct price range he will have the option to view past occupancy history that is maintained by the system.

UC-8 BillPay: (can be considered for future work)

Once the user clicks on the Pay My Bills tab the user will be able to view the current balance that he is responsible for. If the user then clicks on the Pay Now button he/she will be asked the type of payment method (credit or debit) that he would like to use. After that he has to choose the type of card that he would like to use. Then he will need to provide some card details such as name of the cardholder, their mailing address, card number, security code and the expiration date on the card. Once the information is provided and the user clicks on the Finish button the process of bill payment will be complete. The system will then contact a service provider such as PayPal to process the transaction.

Alternate scenario:

For registered customers, a failure to pay the bill will result in the addition of a late fee to their existing bill. Unregistered customers will be allowed to exit if and only if their bill is paid at exit.

ii. Use case diagram

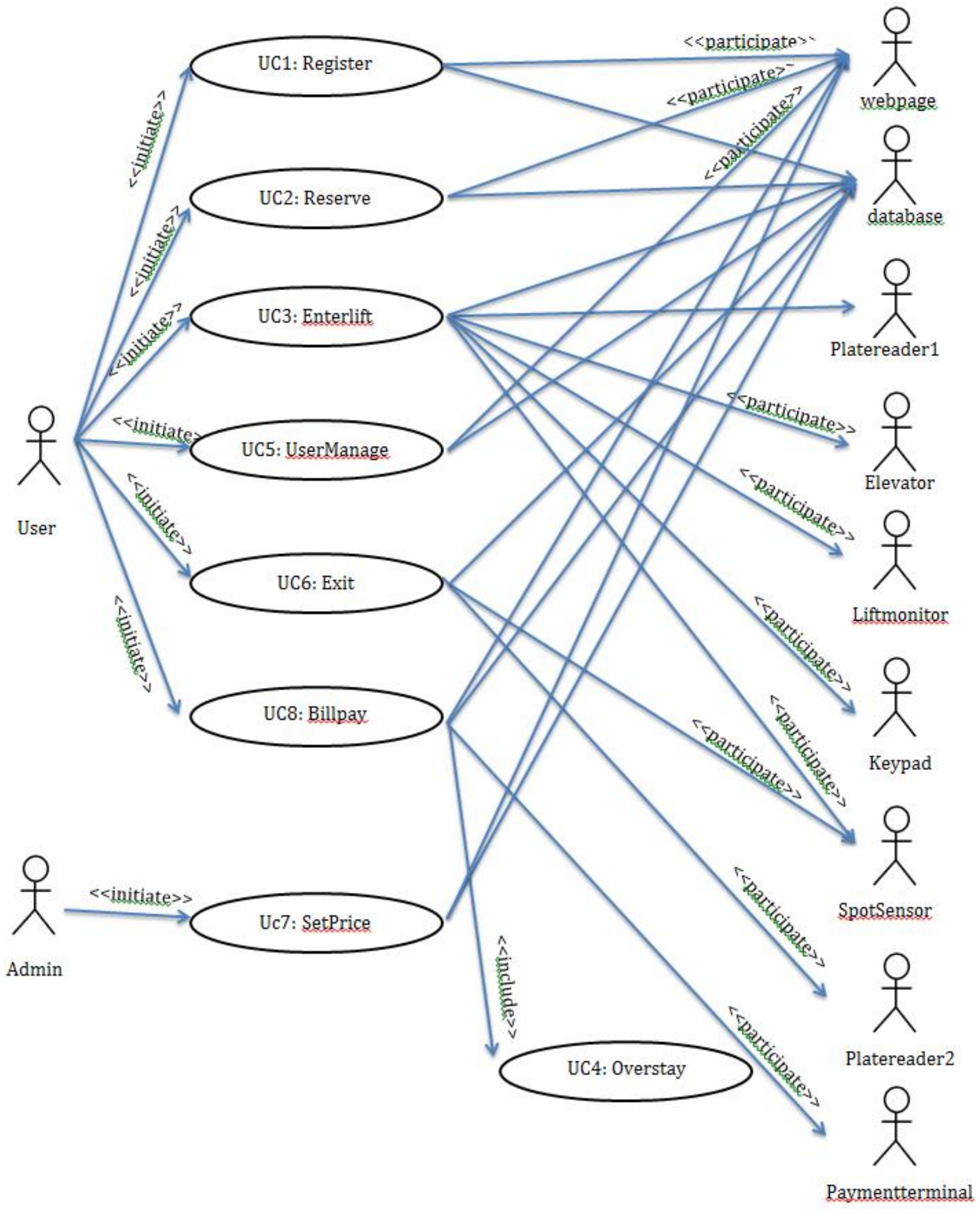


Figure -1

iii. Traceability matrix

Requirement	PW	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7
REQ-1	2	X						
REQ-2	1		X					
REQ-3	1		X					
REQ-4	2							X
REQ-5	5			X				
REQ-6	4			X				
REQ-7	4			X				
REQ-8	4			X				
REQ-9	4			X				
REQ-10	2			X				
REQ-11	4						X	
REQ-12	4					X		
REQ-13	3					X		
REQ-14	3						X	
REQ-15	3							
REQ-16	2						X	
REQ-17	3			X				
REQ-18	4			X				
REQ-19	3			X				
REQ-20	1					X		
REQ-21	2				X			
REQ-22	5		X	X		X		
REQ-23	5	X						
REQ-24	4		X					
REQ-25	3	X	X	X		X		
REQ-26	3	X		X		X		X
REQ-27	4		X	X	X	X	X	
REQ-28	5			X		X		
REQ-29	3		X					
REQ-30	3		X					
MAX PW		5	5	5	4	5	4	3
TOTAL PW		13	24	53	6	28	13	5

iv. Fully-Dressed Description:

Use Case UC-1:	Register
Related Requirements:	REQ-1
Initiating actor:	User
Actor's Goal:	To create an account
Participating Actors:	Database, Webpage
Preconditions:	None worth mentioning. (But note that use case is not available through the elevator display at the garage)
Postconditions:	The user has an account with AUTOmatedPARK and will be considered a registered user.

Flow of Events for Main Success Scenario:

→1. The user goes to the login page and selects to create an account.

←2. The account creation page is displayed to the user and is requested to provide information for the required fields.

→3. The user inputs the required details to complete registration and clicks register.

←4. The system checks to see if all the required details are provided and valid and prompts the user for the correct inputs till it receives a good one.

←5. After all the required valid details are provided the system updates the database and displays a confirmation page to the user.

Use Case UC-2:	Reserve
Related Requirements:	REQ2, REQ3
Initiating actor:	User
Actor's Goal:	To reserve a spot for a fixed period of time
Participating Actors:	Database, Webpage
Preconditions:	The user is a registered user of AUTOmatedPARK
Postconditions:	The user has a reservation for the requested time period if spots are available. The user is advised to check back later if there are no spots available.

Flow of Events for Main Success Scenario:
→1. Customer logs into the system and clicks "Make reservation"
→2. Customer will choose a monthly contract reservation or a more limited-time reservation
Months will be displayed for monthly reservations and dates and times will be displayed for regular reservations
←3. Database successfully checks for free spots during the given time and returns with a rate
→ 4. User will enter a license plate number and agree to terms
←5. Database will be updated with new reservation

Flow of Events for Extensions (Alternate Scenarios):
3. a Database checks for free spots and returns negative
← 1. Customer will be asked to check back later.

Use Case UC-3:	Enter Lift
Related Requirements:	REQ5, REQ6, REQ7, REQ8, REQ9, REQ10, REQ17, REQ18, REQ19
Initiating actor:	User
Actor's Goal:	To park at the garage
Participating Actors:	Database, License Plate reader 1, Elevator, Lift monitor, Keypad, Spot Sensor, Gate
Preconditions:	The user arrives at the garage and enters the elevator.
Postconditions:	The user has parked at the garage if spots are available and user agreed to the terms. The user has left if either there are no available spots or the user does not agree to the terms.

Flow of Events for Main Success Scenario:

←1. The user arrival turns on the license plate reader 1 and it reads a number that is being recognized as a reserved user.

←2. The system checks the database to see if there are spots available.

←3. If there are spots available the user is assigned a spot and the lift monitor displays the assigned spot.

←4. The elevator transports the car to the appropriate floor.

←5. The number of available spots in the floor is updated in the database.

→6. The user parks in the assigned spot.

←7. The spot sensor senses the occupancy and updates the spot occupancy in the database.

Flow of Events for Extensions (Alternate Scenario #1):

3.a. There are no available spots in the upper levels.

←1. The system checks to see if there are spots available in the ground floor.

←2. If there are spots available then the system assigns a spot to the customer. (If there are no spots available the customer will be asked to leave and a rain cheque will be issued).

←3. The lift monitor displays the assigned spot to the user.

←4. The elevator gate opens and lets the customer into the ground floor.

→5. The database will be updated to reduce the number of available spots in the ground floor by one.

→6. The user parks in the assigned spot.

←7. The spot sensor senses the occupancy and updates the database.

Flow of Events for Extensions (Alternate Scenario #2):

1.a. The license plate reader reads a number that is being recognized as a contract user.

←1. The lift monitor displays the spot assigned to the user

←2. The elevator transports the vehicle to the appropriate floor.

→3. The user parks in the assigned spot.

←4. The spot sensor senses the occupancy and updates the spot occupancy in the database.

Flow of Events for Extensions (Alternate Scenario #3):

1.a. The license plate reader reads a number that is not recognized as a reserved or contract customer.

←1. The lift display asks the user to select what type of user he is.

If the user is a reserved user:

←. 1. The lift display asks the user to input the login details to verify the registration and reservation.

→2. The user provides valid registration information.

←3. The system links the license plate number read to the user's current reservation.

Follow the same procedure as a regular reserved customer.

If the user is a registered unreserved user:

←1. The lift display asks the user to input login details to verify the registration.

→2. The user provides valid registration information.

←3. The system checks the database to see if there are spots available in the ground floor.

←4. If there are spots available in the ground floor the lift display asks the user if he agrees to the walkin terms and conditions.

→5. The user agrees to the terms and conditions (If the user does not agree to the terms he will be asked to leave).

←6. The system links the current license plate number to the user's account and assigns and displays the spot to the user.

←7. The gate lets the user into the ground floor and updates the database

→8. The user parks in the assigned spot.

←9. Spot sensor senses occupancy and updates the database.

Flow of Events for Extensions (Alternate Scenarios):

4.a. There are no spots available in the ground floor.

←1. The lift display displays the unavailability of spots and asks the user to leave.

→2. The user leaves the garage.

If the user is an unregistered user:

←1. The system checks the database to see if there are spots available in the ground floor.

←2. If there are spots available in the ground floor the lift display asks the user if he agrees to the walkin terms and conditions.

→3. The user agrees to the terms. (If the user does not agree to the terms he will be asked to leave).

←4. The system creates a temporary account in the database with the license plate number read by the license plate reader.

←5. The system assigns and displays the spot to the user and lets the user to the ground floor.

→6. The user parks in the ground floor.

Flow of Events for Extensions (Alternate Scenarios):

2.a. There are no spots available in the ground floor.

←1. The lift display displays the unavailability of spots and asks the user to leave.

→2. The user leaves the garage.

Use Case UC-4:	Overstay
Related Requirements:	REQ21
Initiating actor:	User
Actor's Goal:	To stay beyond the reservation period without extending the reservation.
Participating Actors:	Database
Preconditions:	The user has a reservation and has parked in the garage during the reservation period.
Postconditions:	The user is notified of the overstay policy.

Flow of Events for Main Success Scenario:

→1. The reserved user who has parked in the garage does not leave the garage by the end of the reservation period.

←2. The system checks the database and finds that the user is overstaying.

←3. The system updates the database to record the remaining duration of stay as overstay.

←4. The system sends a text message to the user warning him of the overstay prices and policy.

Use Case UC-5:	User Manage
Related Requirements:	REQ13, REQ14, REQ20
Initiating actor:	User
Actor's Goal:	To edit user profile or reservations.
Participating Actors:	Webpage, Database
Preconditions:	The user is a registered user and may have reservations.
Postconditions:	The user has successfully modified the user profile or reservations.

Flow of Events for the Main Success Scenario:

→1. A user with a reservation logs into the webpage and selects the "Manage Reservations" option under the Manage tab.

←2. The webpage displays the list of reservations that the user has and prompts the user to select the reservation that he likes to modify.

→3. The user selects the reservation and selects the cancel option.

←4. The system checks to see if the request is being made at least 30 minutes prior to the reservation start time.

←5. a If it is then the reservation is cancelled and the user is notified.

←5.b. If it is not then the reservation will not be cancelled and the user will be notified that he will be charged for the entire reservation period.

Flow of Events for Extensions (Alternate Scenario 1):

3.b. The user selects the reservation that the likes to modify and chooses the extend option.

←1. The user is asked to select the duration for which he would like to extend the reservation.

→2. The user inputs the duration for extension.

←3. The system checks to see if there are available spots during the requested period.

←4.a. If there is availability the system extends the reservation and the webpage notifies the user.

←4.b. If there is no availability the webpage notifies the user that the extension cannot be processed.

Flow of Events for Extensions (Alternate Scenario 2):

1.a. The user selects the "Manage Profile" option under the Manage tab

←1. The webpage displays the list of details that the user provided during registration.

→2. The user inputs data for the fields that he would like to modify.

←3. The system checks the validity of the inputs and updates and displays the changes to the user

Use Case UC-6:	Exit
Related Requirements:	REQ11, REQ16
Initiating actor:	User
Actor's Goal:	To leave the garage.
Participating Actors:	Spot Sensor, Database, License Plate reader 2
Preconditions:	The user has parked in the garage.
Postconditions:	The user has left the garage.

Flow of Events for the Main Success Scenario:

→1. A parked user leaves the spot.

←2. The spot sensor senses that the car has left and updates the spot as available in the database.

←3. If the leaving user is not a contract user, the number of available spots in the particular floor is updated in the database.

→4. The user exits the garage.

←5. The license plate reader 2 reads the plate number of the leaving customer and confirms that the user has exited.

Use Case UC-7:	SetPrice
Related Requirements:	REQ4
Initiating actor:	Owner
Actor's Goal:	To set parking prices for the garage users.
Participating Actors:	Webpage, Database
Preconditions:	The owner has administrative privileges.
Postconditions:	The owner has set the desired parking prices for the garage.

Flow of Events for Main Success Scenario:

→1. The owner logs into the webpage with administrative privileges.

→2. The owner clicks the "View History" tab.

←3. The webpage displays the garage usage history.

→4. The owner selects to set prices for the garage.

←5. The webpage displays the current price values for the various kinds of users.

→6. The owner sets the new parking prices.

←7. The system checks for the validity of the information provided.

←8. The new prices are updated and the owner is notified of the changes.

D. System Sequence Diagrams

System sequence diagrams for implemented use cases.

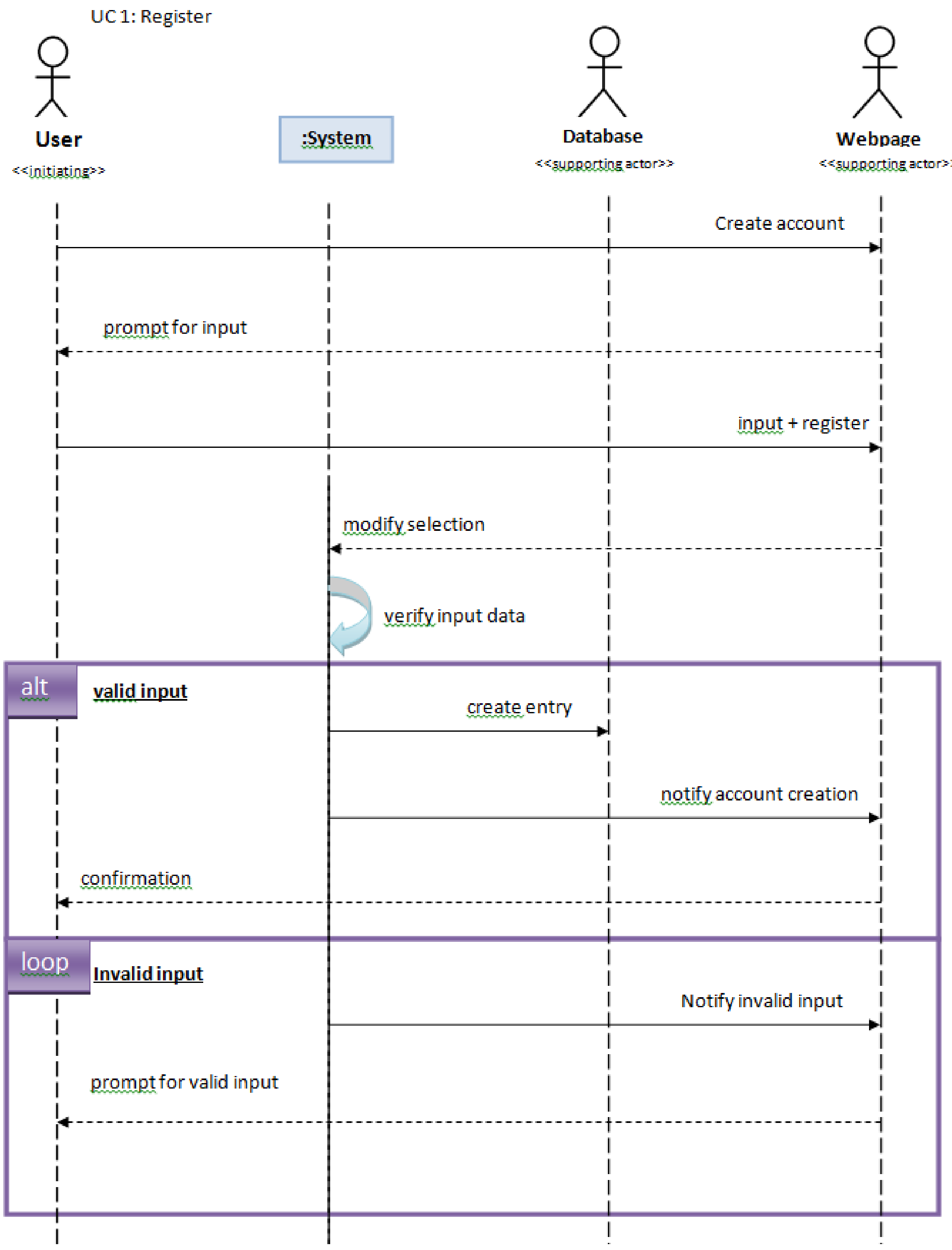


Figure-2

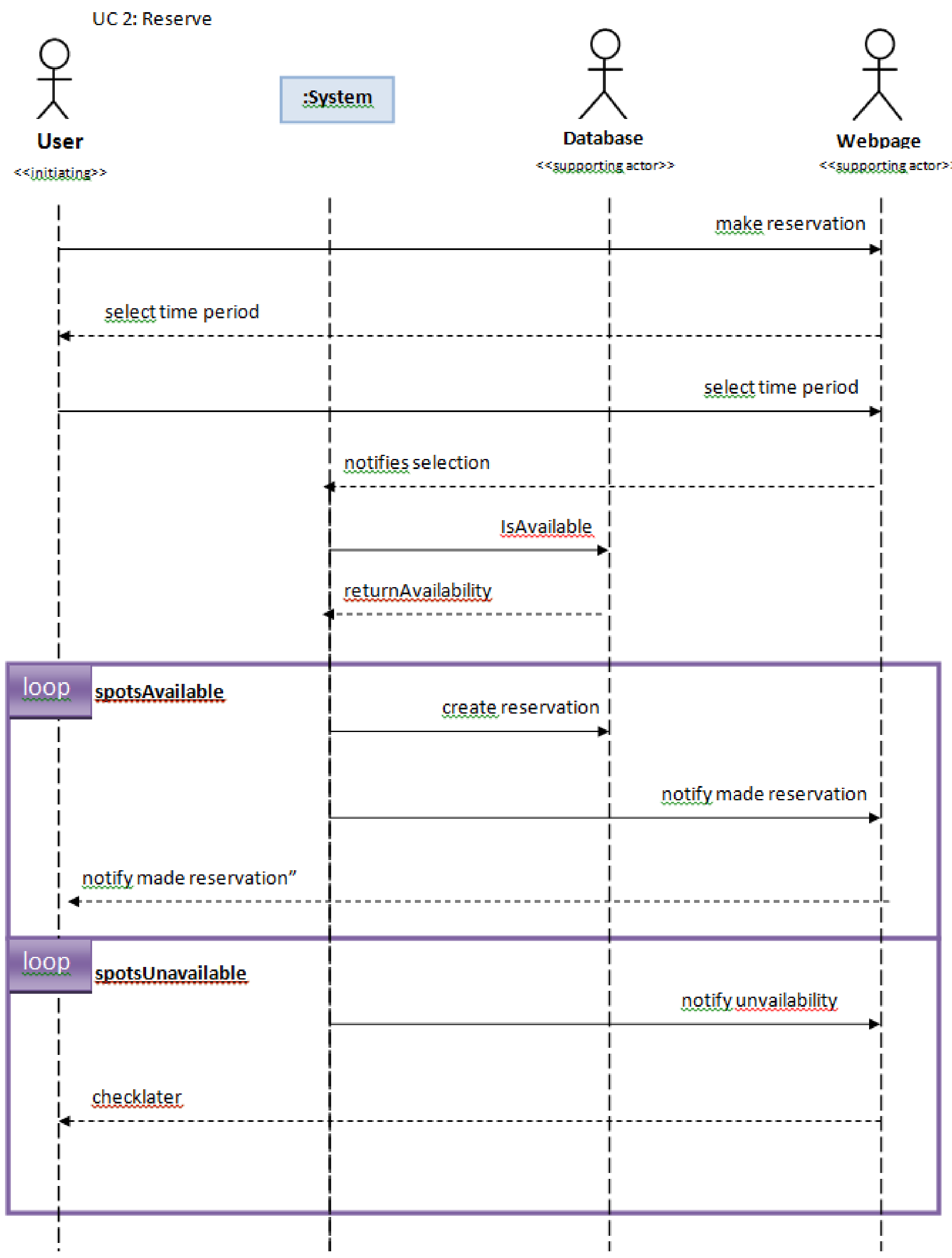

UC 2: Reserve
User
<<initiating>>
:System
Database
<<supporting actor>>
Webpage
<<supporting actor>>
make reservation
select time period
select time period
notifies selection
IsAvailable
returnAvailability
loop
spotsAvailable
create reservation
notify made reservation
notify made reservation"
loop
spotsUnavailable
notify unvailability
checklater

Figure-3

Uc-3: Enterlift

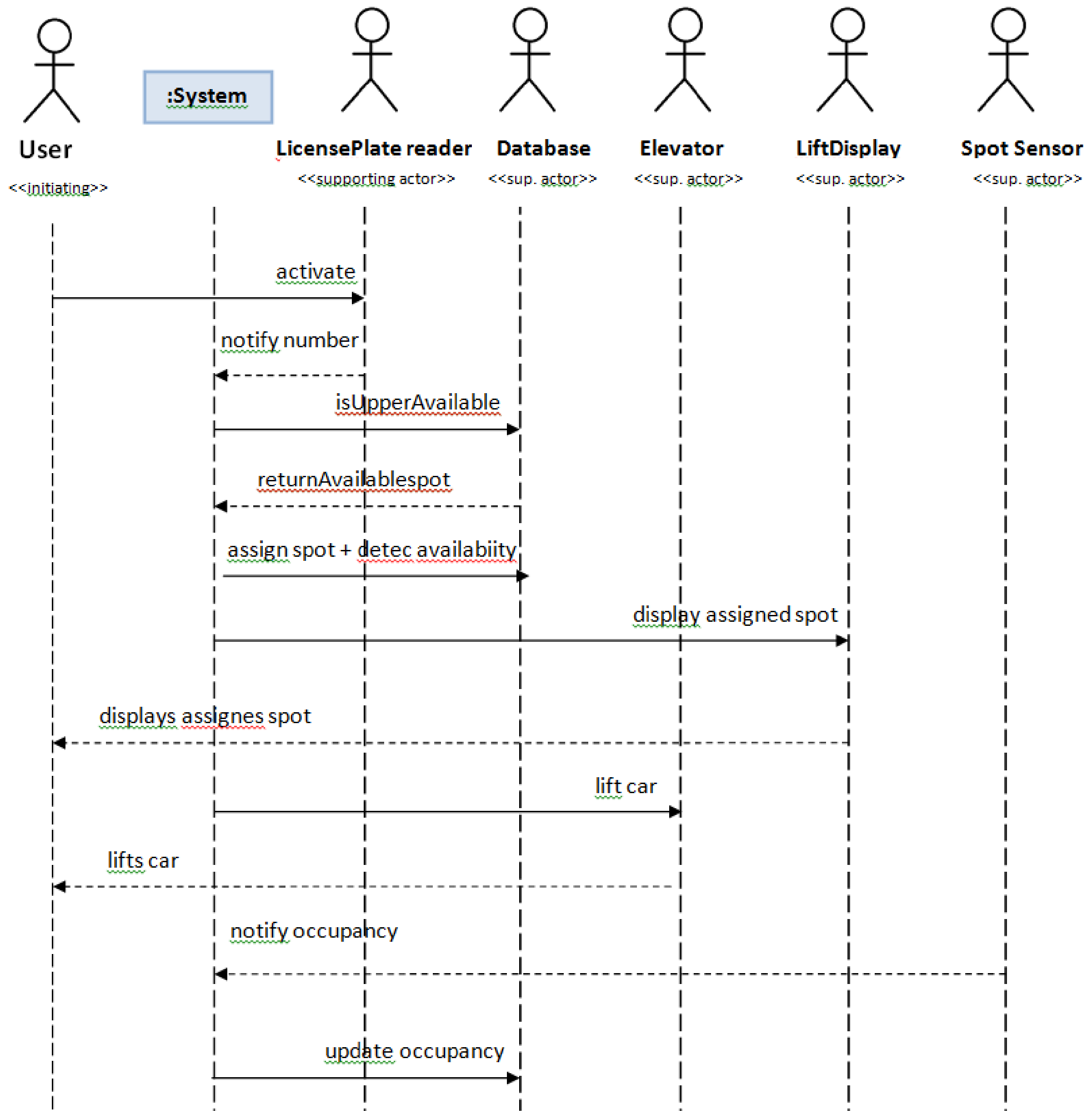


Figure-4

Alternate scenario 1

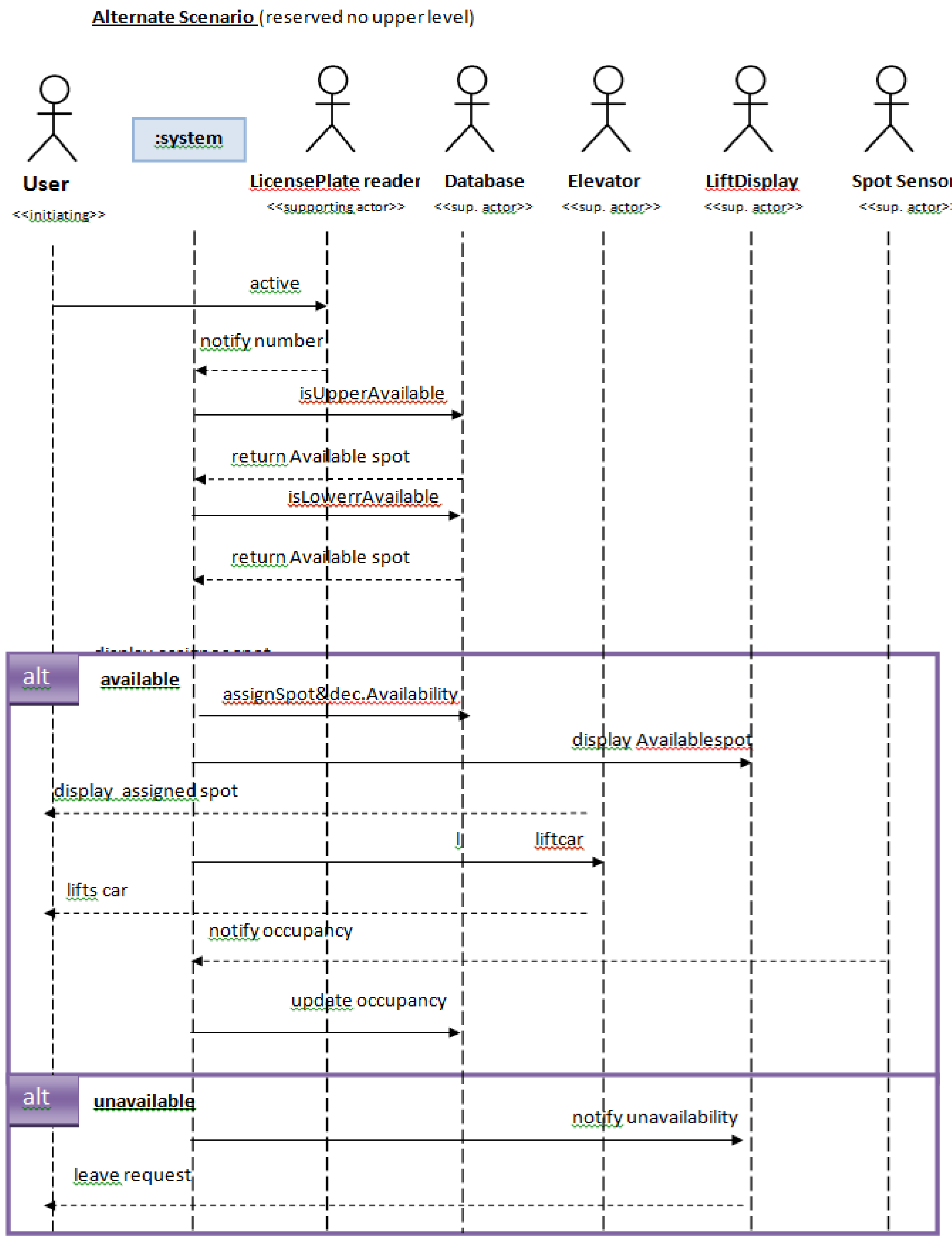


Figure-5

Alternate Scenario 2

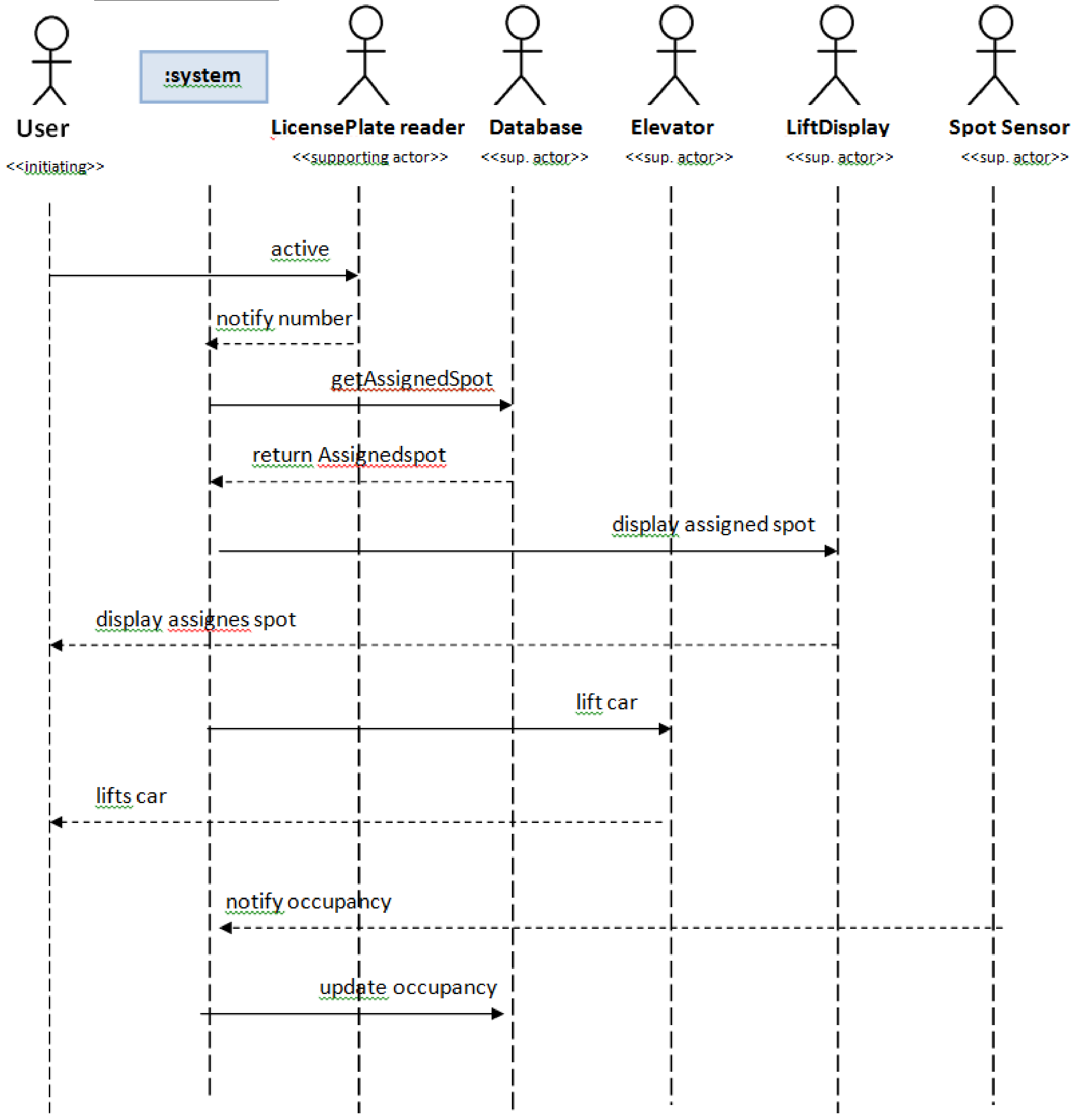


Figure-6

Alternate Scenario 3 (plate not recognized)

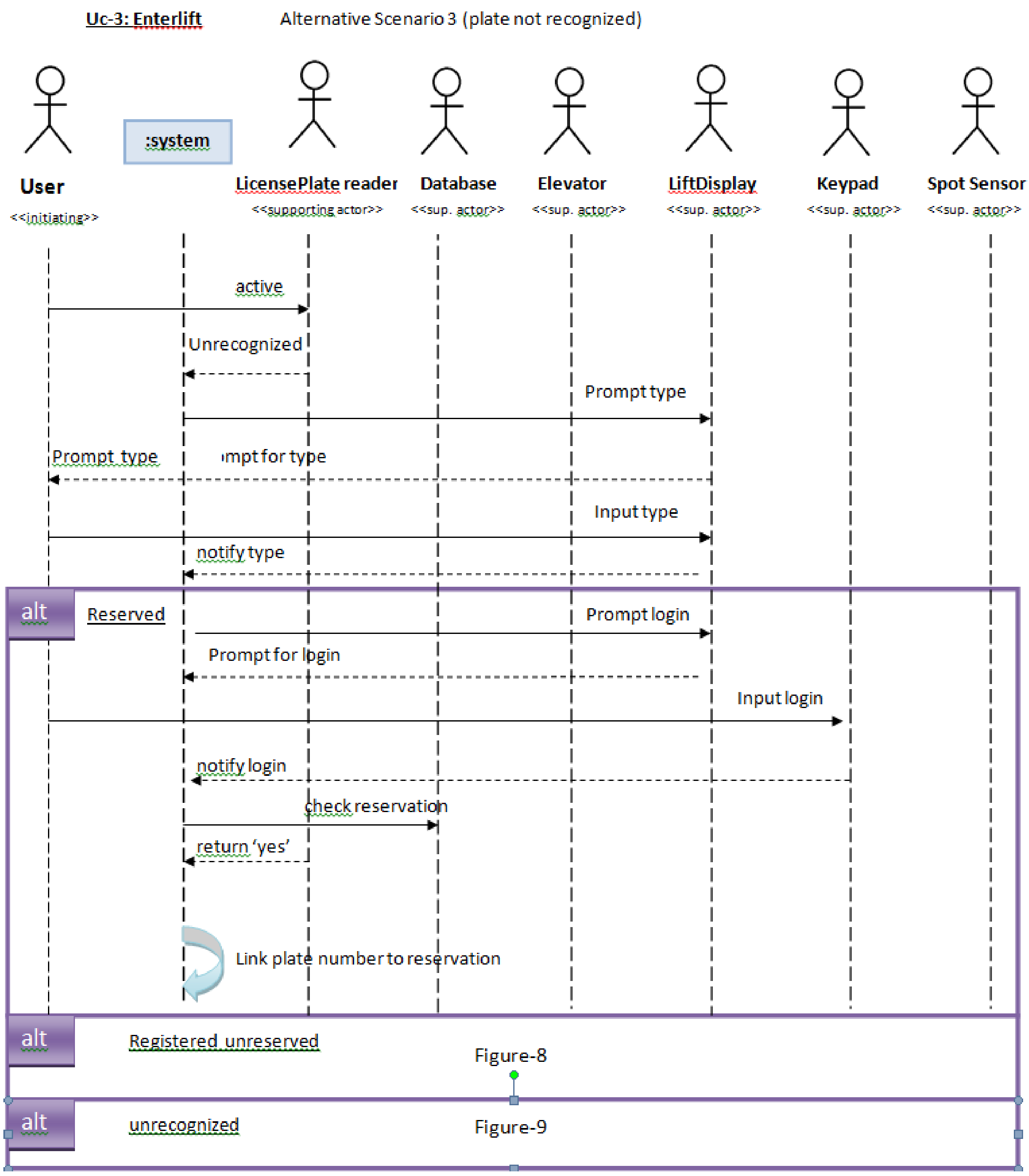


Figure-7

Alternate Scenario 3 (registered, unreserved)

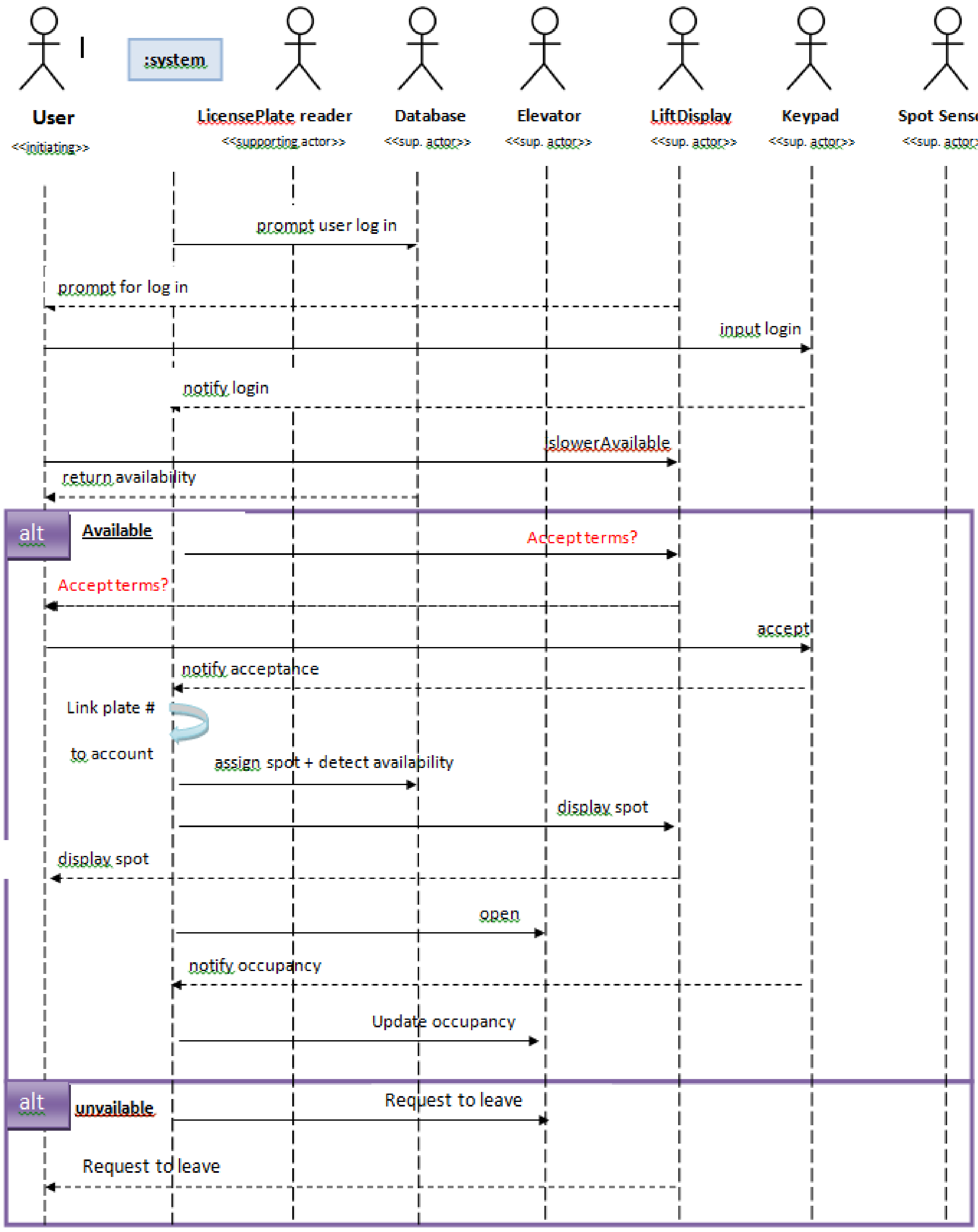


Figure-8

Alternate Scenario 3 (unregistered)

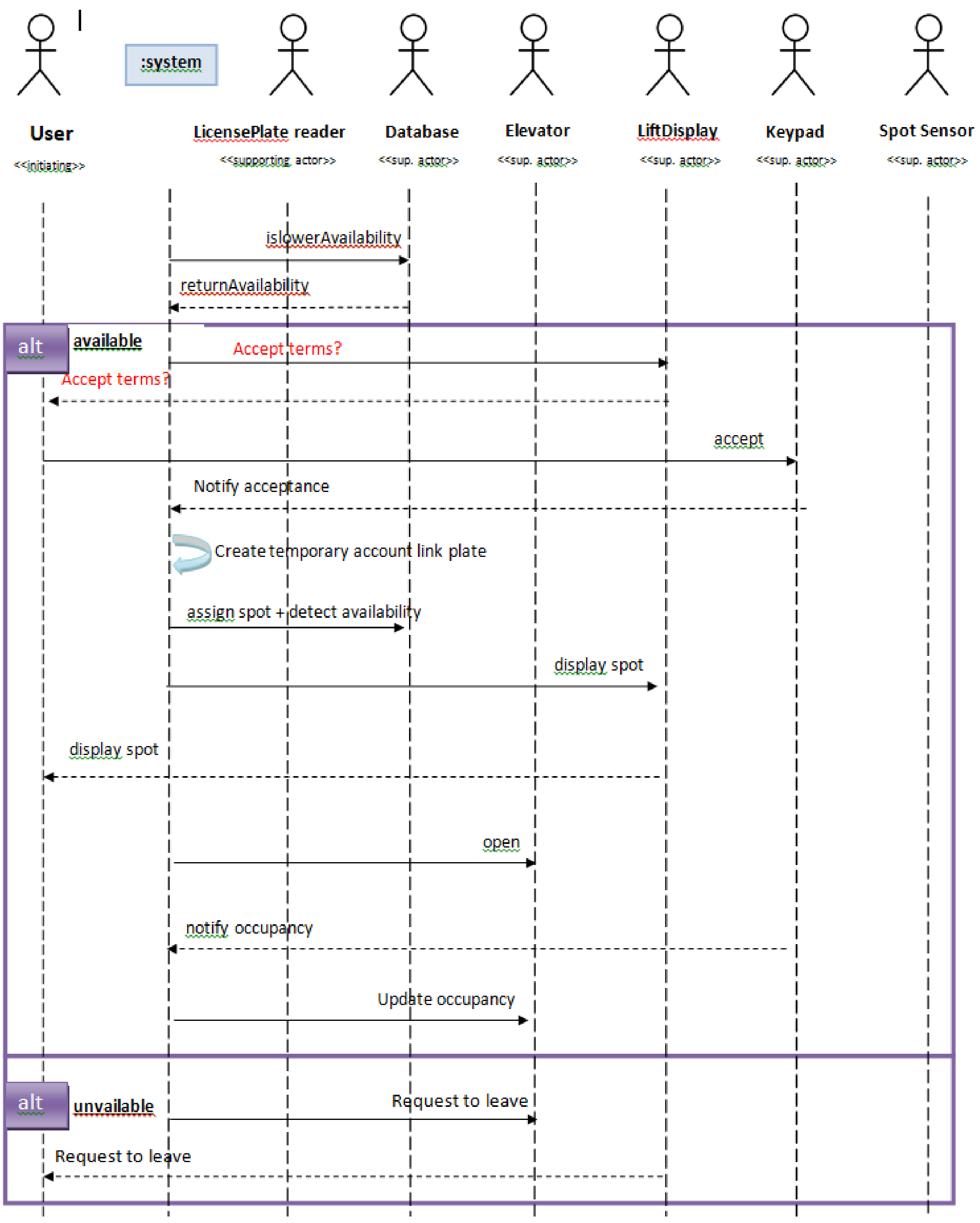


Figure-9

UC-4 OverStay

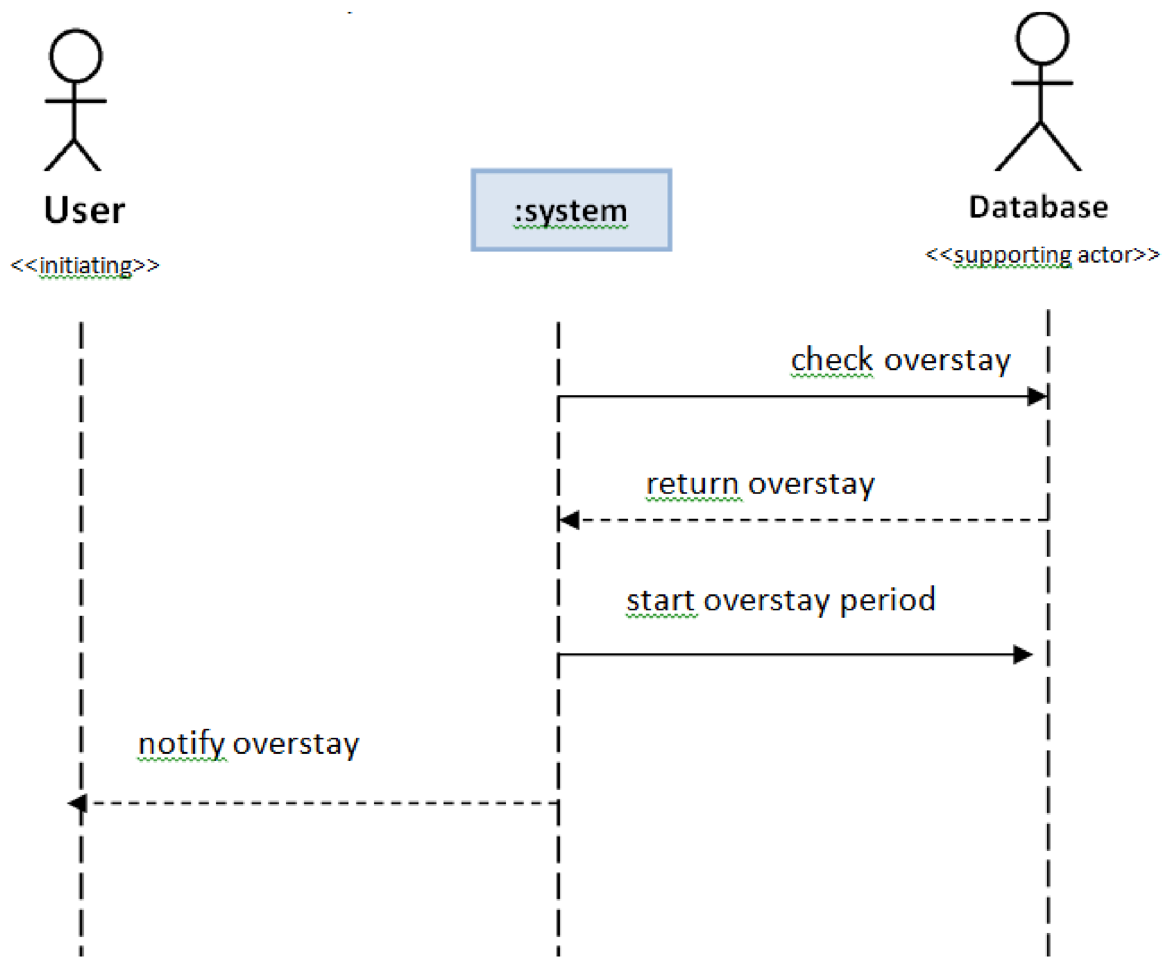


Figure-10

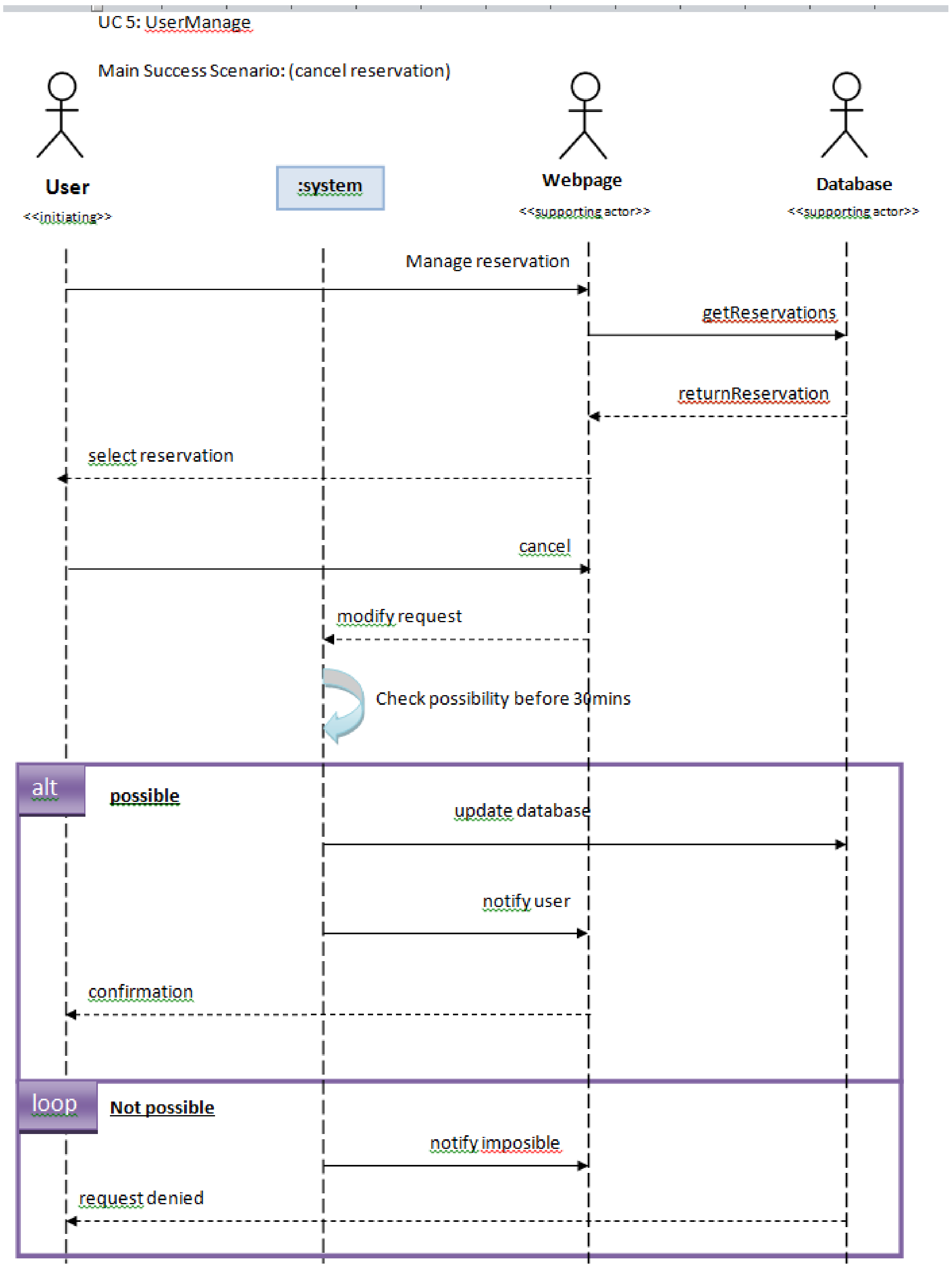

UC 5: UserManage
Main Success Scenario: (cancel reservation)
User
<<initiating>>
:system
Webpage
<<supporting actor>>
Database
<<supporting actor>>
Manage reservation
getReservations
returnReservation
select reservation
cancel
modify request
Check possibility before 30mins
alt
possible
update database
notify user
confirmation
loop
Not possible
notify imposible
request denied

Figure-11

Alternate Scenario 1

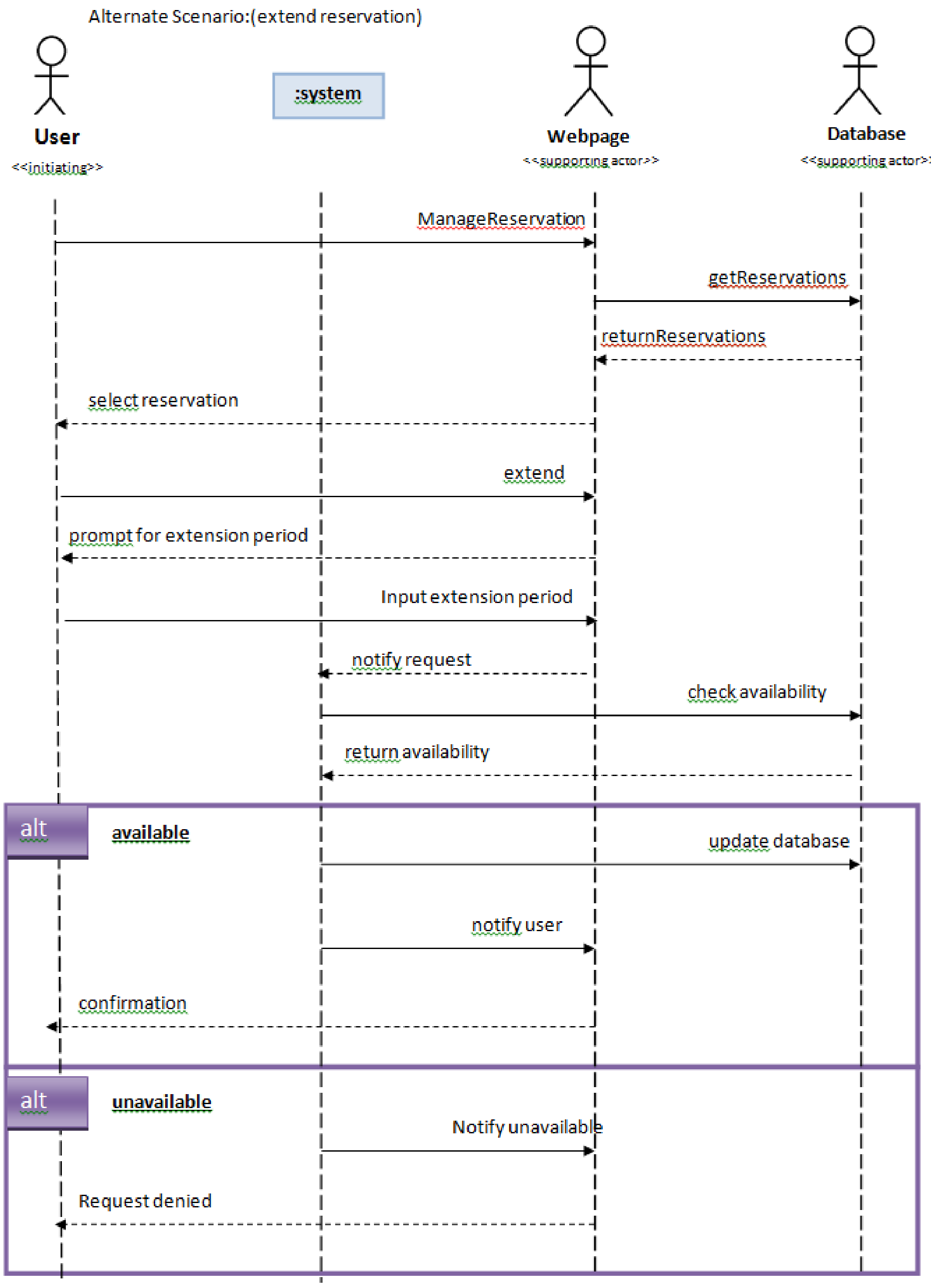


Figure-12

Alternate Scenario 2

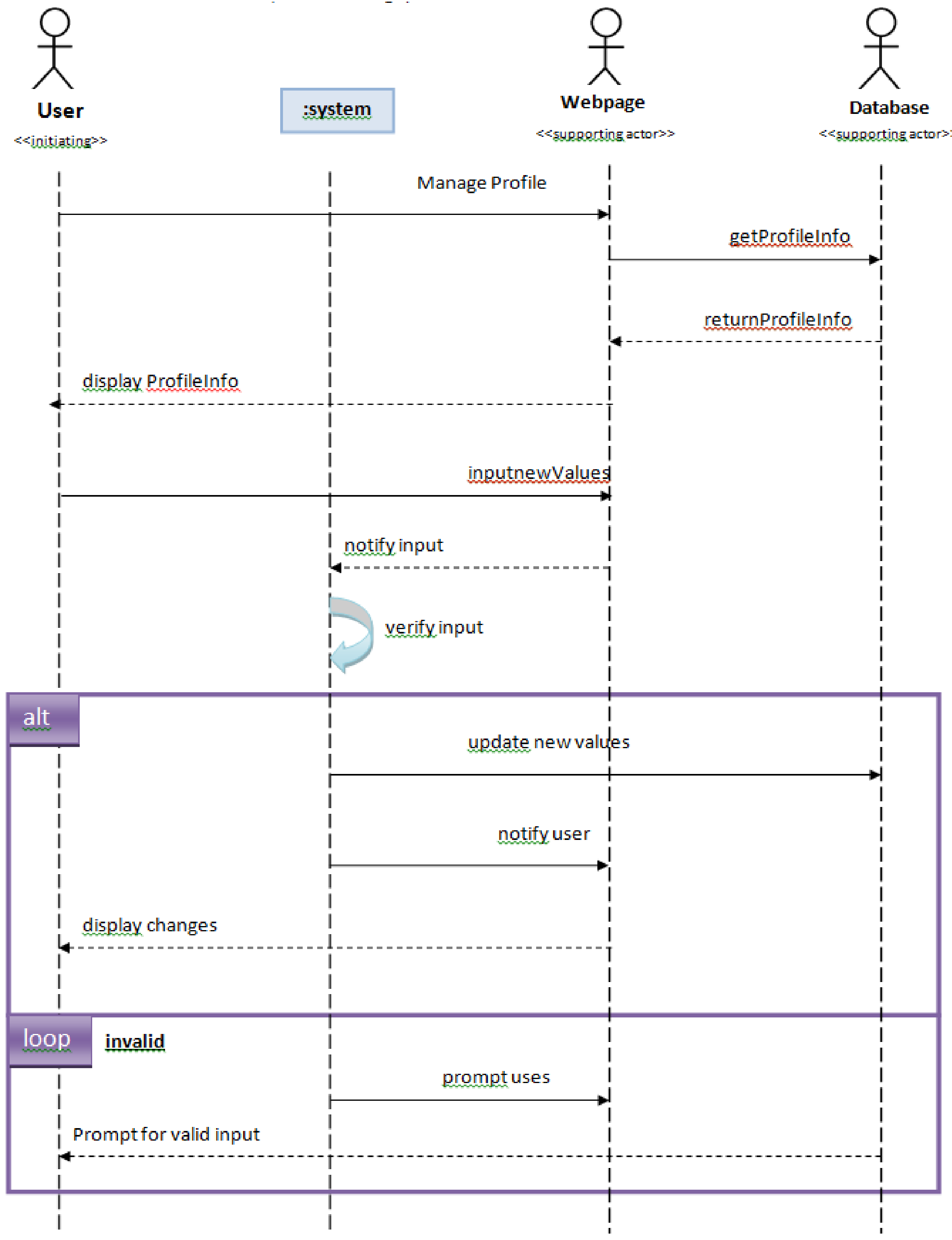


Figure-13

Uc-6: (exit)

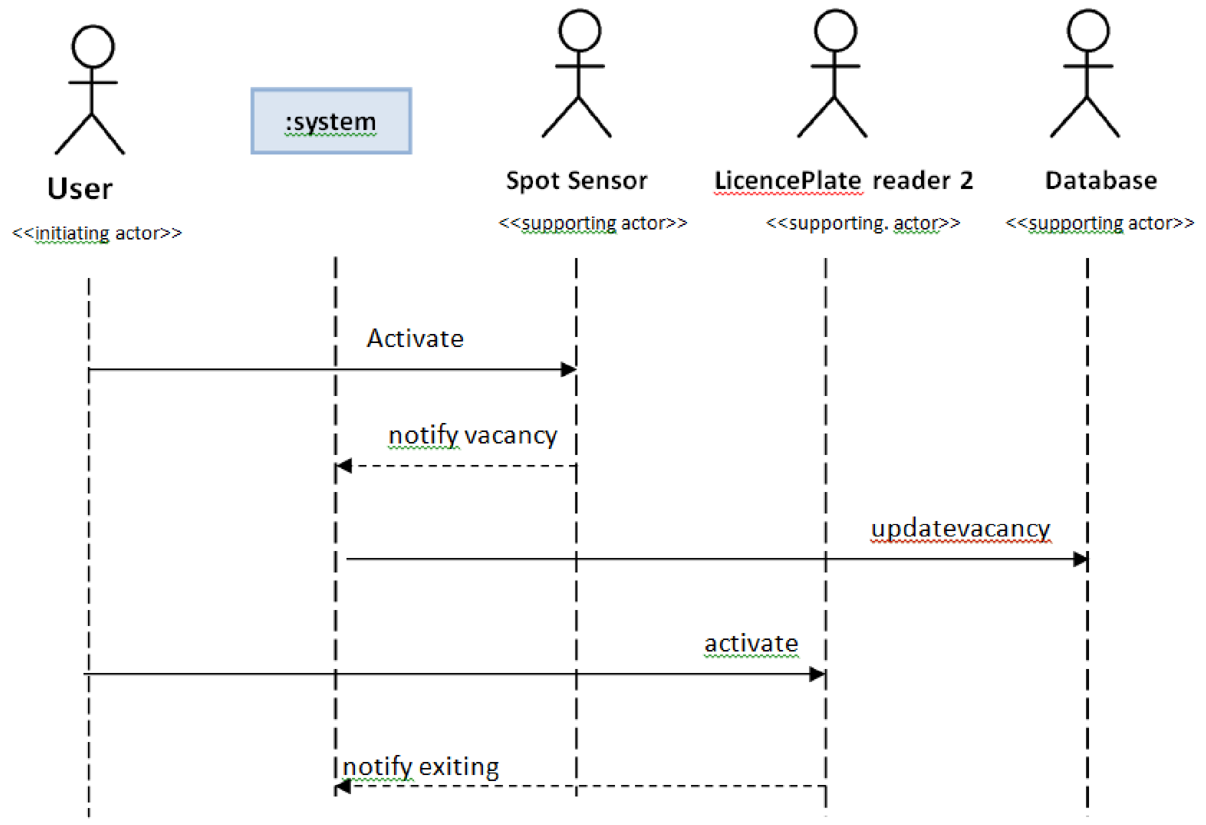


Figure-14

UC 7: SetPrice

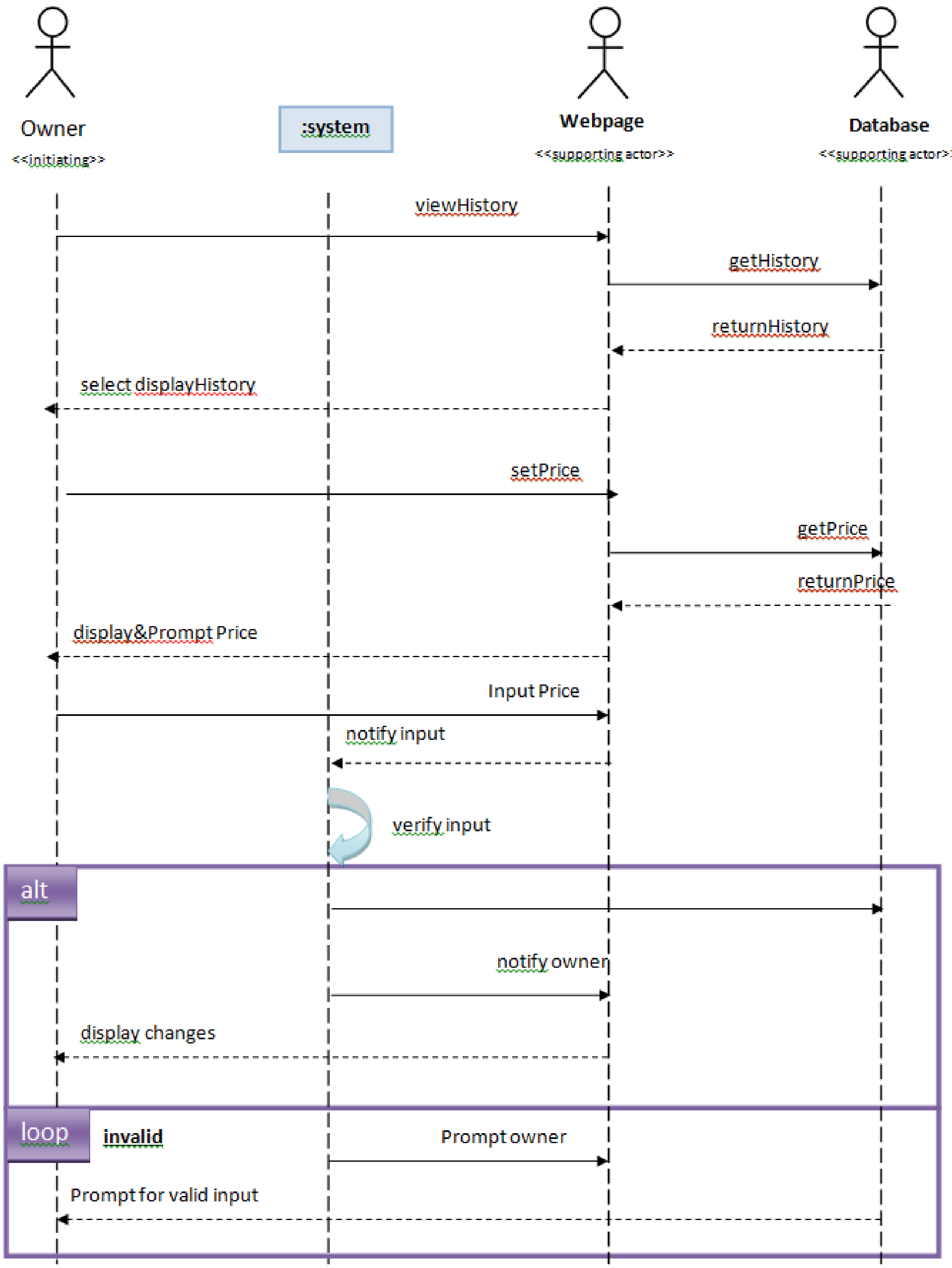


Figure-15

6. Effort Estimation using Use Case Points

1. Unadjusted Actor weight (UAW) calculation

Actor	Factor
Owner	1
User	3
License plate Reader 1	1
License plate Reader 2	1
Lift Monitor	1
Keypad	1
Elevator	1
Payment Terminal	1
Spot Sensors	2
Operator	2
Maintenance Manager	2
Gate/Barrier	1
Webpage	2
Database	3
UAW	22

2. Unadjusted Use Case Weight (UUCW) calculation

Use Case	Type	Weight
UC1	Simple	5
UC2	Simple	5
UC3	Very Complex	20
UC4	Simple	5
UC5 -	Average	10
UC6 -Exit	Complex	15
UC7 -SetPrice	Simple	5
UC8 - BillPay	Average	10
UUCW		75

3. Unadjusted Use case Points (UUCP) calculation
   UUCP = UAW +UUCW = 22+75 = 97

4. Technical complexity factor

Factor	Description	Assigned value	Weight	Extended Value
T1	Test Tools	2	2	4
T2	Webpage system	2.5	3	7.5
T3	Test Environment	2	2	4
T4	Security features	0	2	2

T5	Performance Objective	1	2	2
T6	Distributed system	2	3	6
T7	Complex interfacing	1	2	2
T8	Development Environment	2	3	6
T9	Documented input	4	3	12
			Total	45.5

TCF = 0.6+0.01*45.5=1.055

5. Environment Complexity Factors

Environment factor	Description	Factor	Weight	Extended value
E1	Programming skills	3	2	6
E2	Familiarity with UML	2	1.5	3
E3	Knowledge of the course	4	2	8
E4	Meeting time	2	-1	-2
E5	Mathematical background	1	1.5	1.5
E6	Report writing skills	3	2	6
E7	Cooperative	3	1.5	4.5
			Total	27

ECF = 1.4+0.03* 27=0.59

6. Use case points (UCP) calculation:

UCP = UUCP x TCF x ECF = 97 x 1.055 x0.59 = 60.37

7. Domain Analysis

A. Domain Model

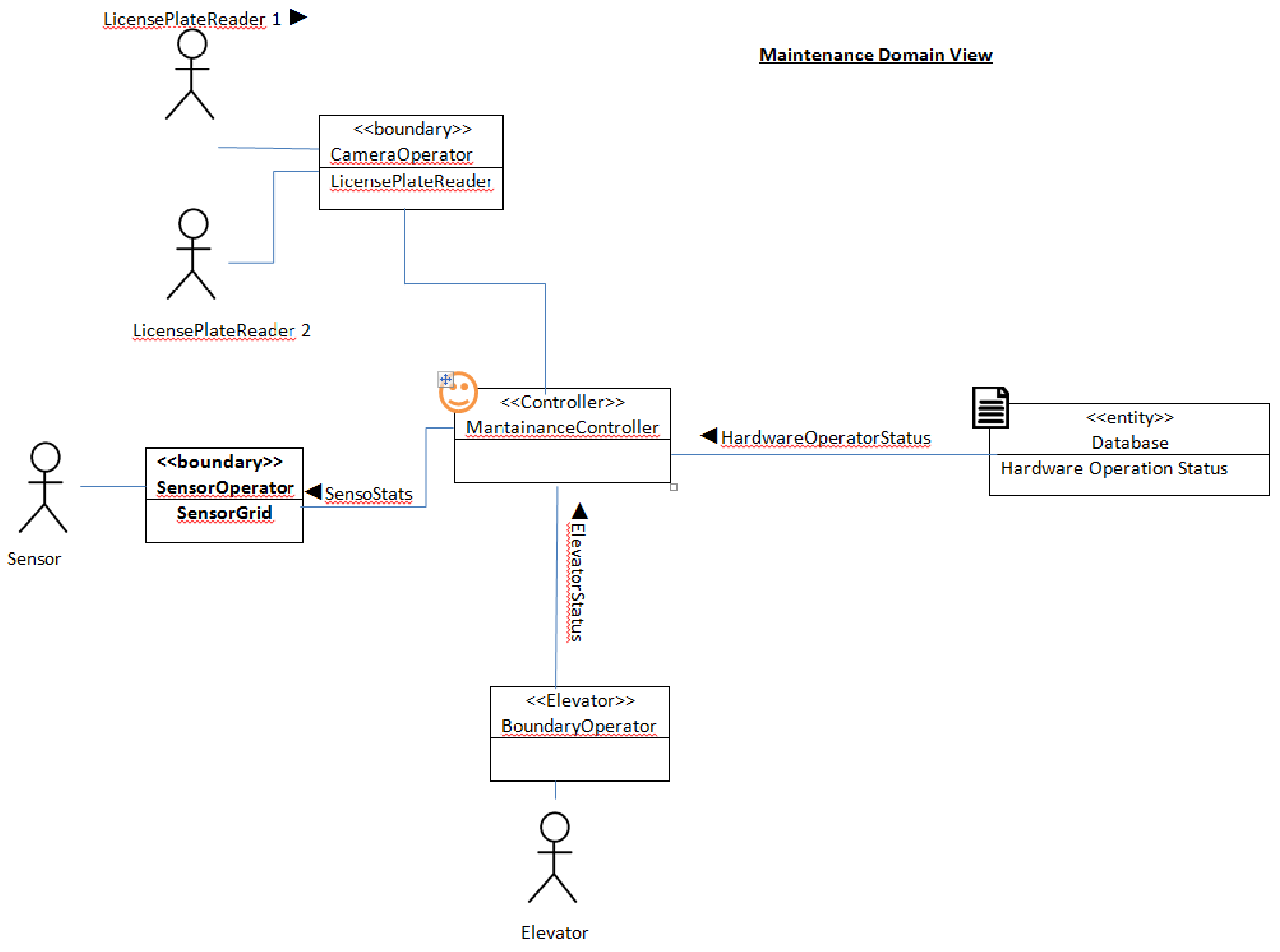
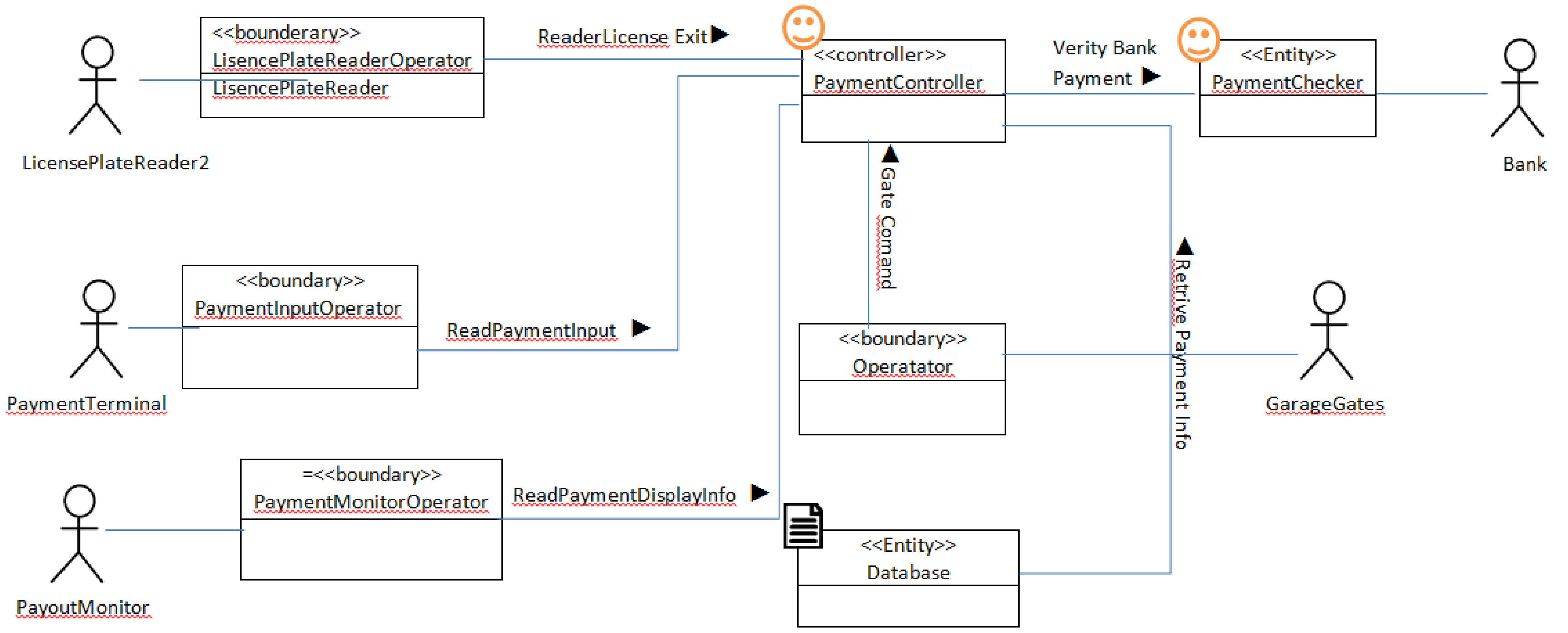
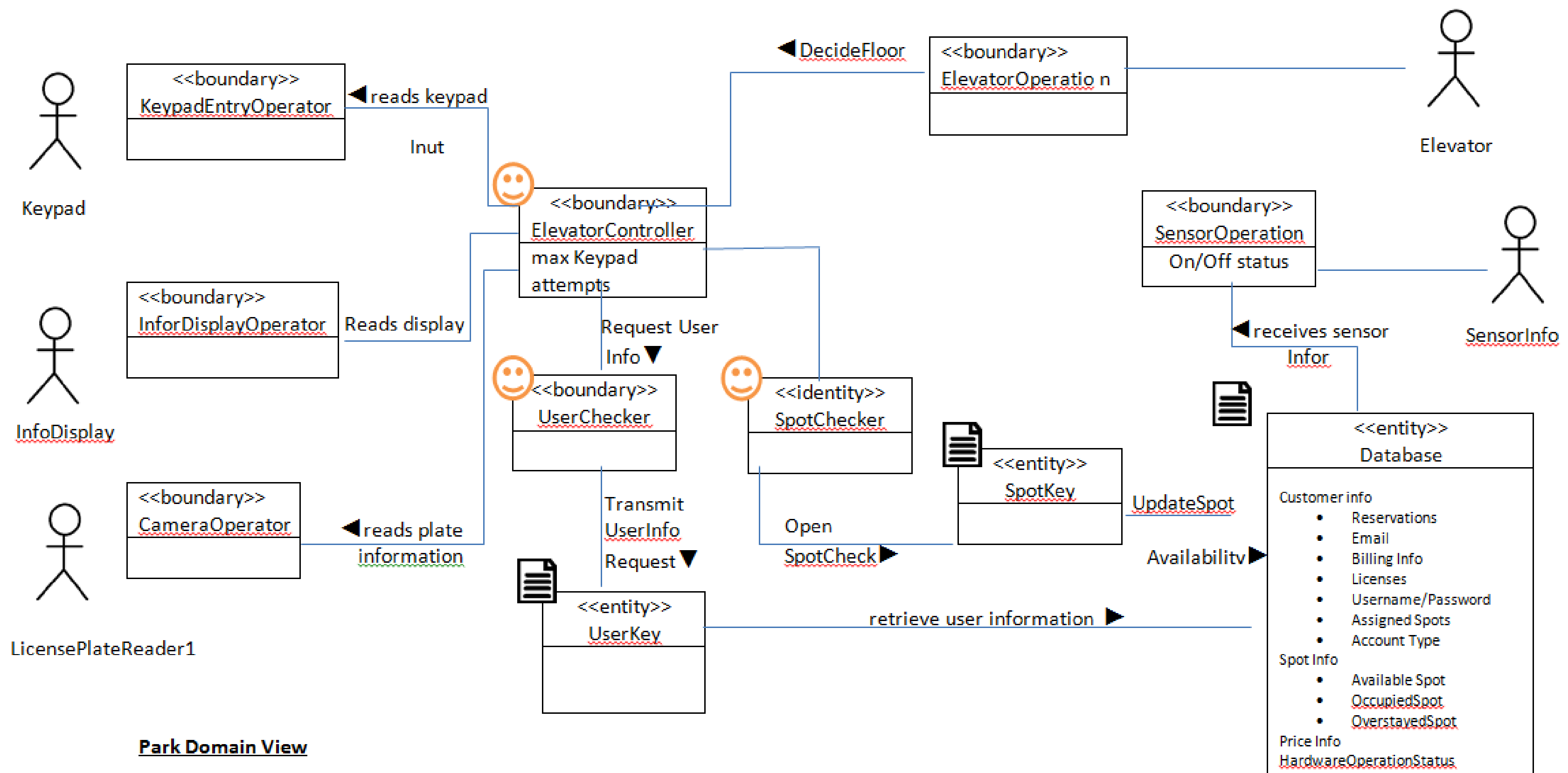


Figure-16



Exit Domain View

Figure-17



<<boundary>>
KeypadEntryOperator
reads keypad
Inut
Keypad
DecideFloor
<<boundary>>
ElevatorOperatio n
Elevator
<<boundary>>
ElevatorController
max Keypad attempts
<<boundary>>
InforDisplayOperator
Reads display
InfoDisplay
<<boundary>>
SensorOperation
On/Off status
SensorInfo
receives sensor Infor
Request User Info
<<boundary>>
UserChecker
<<identity>>
SpotChecker
<<entity>>
SpotKey
UpdateSpot
Availabilitv
<<entity>>
Database
Customer info
Reservations
Email
Billing Info
Licenses
Username/Password
Assigned Spots
Account Type
Spot Info
Available Spot
OccupiedSpot
OverstayedSpot
Price Info
HardwareOperationStatus
<<boundary>>
CameraOperator
reads plate information
LicensePlateReader1
Transmit UserInfo Request
Open SpotCheck
<<entity>>
UserKey
retrieve user information
Park Domain View

Figure-18

Web Domain view

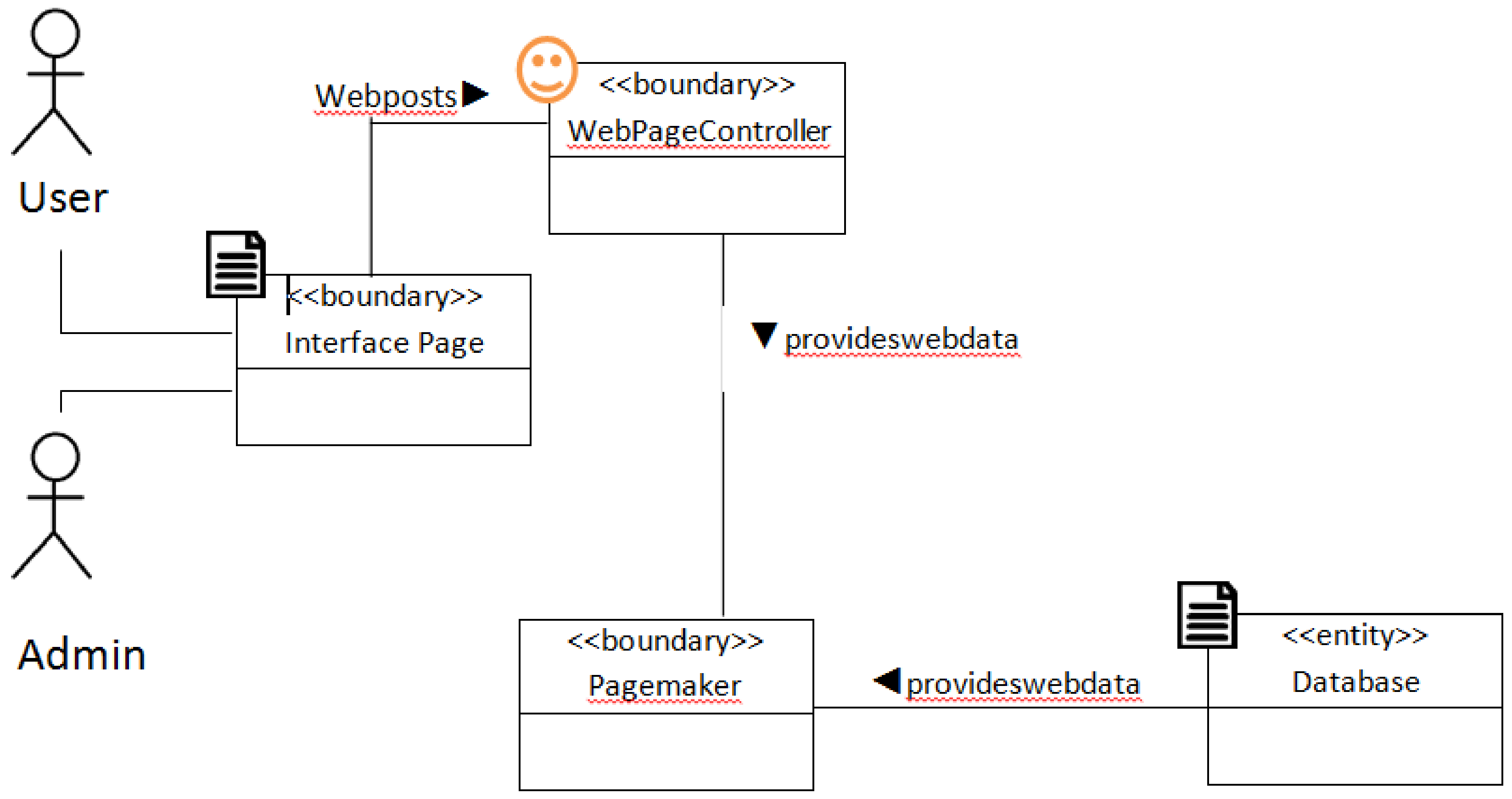


Figure-19

i. Concept definitions

Key: K- Knowing D- Doing

Responsibility Description	**Type**	**Concept Name**
Reads inputs from the sensors to determine occupancy of each spot and informs the SensorChecker. If contradicting inputs are read in from particular spots dual sensors, a signal will be sent to the MaintenanceLogger to record the problem.	K	SensorOperator
Reads input from SensorOperator to update the FreeSpotKey.	D	SensorChecker
Reads input from KeypadEntryOperator, LicensePlateReaderOperator and UserChecker. It outputs prompts and information that it calculates to the User through the Lift Monitor Operator. It determines the appropriate floor to the ElevatorOperator with which to admit the customer (or possibly denies the customer in the event of full-occupancy). It also updates the FreeSpotKey of the User's entry through the UserChecker to be recorded for payment and to enter a reduction in the available number of spots so as to resolve latency issues caused by waiting for the number to be updated when the User reaches their spot sensor.	D	ElevatorController
Outputs commands to the Elevator (Lift).	D	ElevatorOperator
Reads input from the License Plate Readers and communicates with the ElevatorController for arrivals and the PaymentController for exits.	D	LicensePlateReaderOperator
Used to temporarily store all user info that it retrieved from the database for the UserChecker until the User is processed.	K/D	UserKey
Serves to interact between the PaymentController and the UserKey to confirm session info has been logged and payment has been appropriated so the user can exit.	D	PaymentChecker
Receives request from the ElevatorController to obtain the type of customer that has arrived. It then accesses UserKey to determine the type of customer from the profiles where it will start a log for the session. It will also	D	UserChecker

create a Walk-In profile in the UserKey if the customer is not registered for a spot.		
Receives input from the Keypad which is then sent to the ElevatorController to be confirmed.	D	KeypadOperator
Container for current number of parking spots available on each floor and where their subsequent location is, collaborates with the Database.	K	FreeSpotKey
Receives notifications from hardware in the event of an abnormality, sends notifications to local service center and logs information in the Database	D	MaintenanceLogger
Operates exit gate to allow people to exit only after it receives command from the PaymentController	D	GateOperator
Receives input from LicensePlateReader of user approaching to exit. This plate number is then sent to the PaymentChecker to determine amount owed. Once payment criteria is confirmed through the PaymentChecker, the PaymentController will tell the GateOperator to open the gate.	D	PaymentController
Reads the payment received from the Payment terminal in the event a customer must pay before being allowed to exit and sends it to the Payment Controller.	D	PaymentOperator
It will output pertinent information for the User from the ElevatorController through the Lift Monitor.	D	LiftMonitorOperator
Intermediary betw	D	WebPageController
Gathers information to be displayed on the interface	D	PageMaker
Acts as the GUI between the system and the User	K	InterfacePage
Stores all pertinent variable information of the system	K	Database

ii. Association definitions

Concept Pair	Association Description	Association Name
SensorChecker ↔ SensorOperator	SensorChecker obtains sensor readings from SensorOperator in order to update spot availability (FreeSpotKey)	ReadSensor
ElevatorController ↔ SpotChecker	ElevatorController obtains data from SpotChecker to check for spot availability (from FreeSpotKey)	RequestSpotInfo
ElevatorController ↔ ElevatorOperator	ElevatorController sends command to ElevatorOperator to move elevator and waits for execution confirmation	DecideFloor
LicensePlateOperator ↔ ElevatorController	ElevatorController reads information from LicensePlateOperator as a User pulls up to park.	ReadsPlateInfo
KeypadOperator ↔ ElevatorController	ElevatorController reads information from KeyPadOperator as a User pulls up to park if they are a registered user with an unrecognized license plate.	ReadsKeypadInput
ElevatorController ↔ UserChecker	ElevatorController keeps tally of invalidated key inputs that it receives back from UserChecker before they the User is considered a walk-in.	RequestUserInfo
PaymentController ↔ LicensePlateOperator	PaymentController reads license number from LicensePlateOperator as Users pull up to attempt to exit.	ReadExitLicense
PaymentController ↔ PaymentOperator	If it is a walk-in customer or registered customers decide to pay before exiting, the PaymentOperator will send the payment info that it receives to the PaymentController.	ReadsPaymentInput
PaymentController↔ PaymentMonitorOperator	PaymentController will send the amount due to the PaymentMonitorOperator	ReadPaymentDisplay
UserChecker ↔ UserKey	UserChecker asks the UserKey to retrieve any User info that is	TransmitUserInfoRequest

	saved based on its information.	
UserKey ↔ Database	UserKey loads any info pertaining to the request that is found in the Database	RetrieveUserInfo
PaymentController ↔ PaymentChecker	PaymentController sends the license info of who is attempting to exit to the PaymentChecker. The amount owed that is returned is then sent to be displayed on the PaymentMonitor. Any payment that is received from the Payment Operator is verified by the PaymentChecker before the User can exit. If it is a registered customer the PaymentChecker merely records the time that the User has left so that they may be billed later unless they chose to pay before exiting.	VerifyPayment
LiftMonitorOperator ↔ ElevatorController	LiftMonitorOperator displays information it receives from the ElevatorController.	ElevatorDisplay
MaintenanceLogger ↔ SensorOperator	MaintenanceController receives notifications from SensorOperator in the event of any malfunction	SensorStatus
MaintenanceLogger↔ LicensePlateOperator	MaintenanceController receives notifications from LicensePlateOperator in the event of any malfunction	LicensePlateReaderStatus
MaintenanceLogger↔ ElevatorOperator	MaintenanceController receives notifications from ElevatorOperator in the event of any malfunction	ElevatorStatus
MaintenanceLogger ↔ Database	Updates hardware statuses in the database	HardwareOperationStatus
PaymentChecker ↔ PaymentKey	PaymentChecker will receive the amount due from PaymentKey and will update it if payment is received	RequestPaymentInfo
PaymentKey ↔ Database	PaymentKey accesses info based on the info it receives	GetAmountDue

	from PaymentChecker from the Database	
SpotChecker ↔ FreeSpotKey	SpotChecker consults the FreeSpotKey to check for available spots	OpenSpotCheck
SensorChecker ↔ FreeSpotKey	FreeSpotKey receives the input from the sensors through the SensorChecker	UpdateSpotKey
FreeSpotKey ↔ Database	FreeSpotKey and Database keep each other synchronized	UpdateSpotAvailability
PaymentController ↔ GateOperator	PaymentController will send the signal to the GateOperator to lift the gate once payment is confirmed to have been appropriated.	ExitSuccess
WebpageController ↔ InterfacePage	WebpageController receives info from the InterfacePage and sends request to the PageMaker to build a response	WebPosts
InterfacePage ↔ PageMaker	InterfacePage reads the information collected by the PageMaker	PreparesWebPage
PageMaker ↔ database	PageMaker queries database for pertinent information	providesWebData
PageMaker ↔ WebPageController	PageMaker receives the request from the WebPageController	ConveysWebRequests

iii. Attribute definitions

LoginAttempts	Number of unsuccessful keypad entries
deviceName	Name of the hardware with the error
LicenseNumber	String of characters that represent the license plate information taken from the License Plate Readers
errorMessage	String pertaining to the type of error with the hardware
passWord	…
userID	…
ReservationData	Time slot to hold a users reservation time
SessionOwed	Amount owed that is taken every parking session unless they are a guaranteed customer during the extent of their contract time
TotalOwed	Total of all SessionOwed-SessionPaid
SessionPaid	Total amount paid per session
DoorStatus	present status of elevator door position
ExitGateStatus	present status of exit gate position
MaintenanceIssue	String to log into MaintenanceKey
GuaranteedData	Time slot to hold a guaranteed users contract times with which to guarantee a spot
FreeSpots	Bitmap of spot availability
CreditData	Record of credit card information if the user opts to provide it to us rather than receiving a monthly bill
OpenSession	Time entered and pending duration of current stay

iv. Traceability Matrix

USE CASE	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7
PW	13	24	53	6	28	13	5
Domain Concept							
SensorOperator				X		X	
SensorChecker				X		X	
ElevatorController			X				
ElevatorOperator			X				
LicensePlateReaderOperator			X				
UserKey			X				
PaymentChecker							
UserChecker			X				
KeypadOperator			X				
FreeSpotKey			X	X			
MaintenanceLogger							
GateOperator						X	
PaymentController						X	
PaymentOperator						X	
LiftMonitorOperator			X				
WebPageController	X	X			X		X
PageMaker	X	X			X		X
InterfacePage	X	X			X		X
Database	X	X	X	X	X	X	X

B. System Operation Contracts

Operation	Park
Preconditions	• FreeSpots>=1 and they are not reserved for someone else • licenseNumber has been confirmed with a user profile or entered into Walk-In profile • front DoorStatus is has been open to allow customer into lift for assessment while back door of lift has remained closed to prohibit premature entry. • Front OpenDoor has been closed to safely transport the car and backDoorStatus has been open to allow car into garage. • ReservationData is checked and confirmed if applicable to customer • GuaranteedDate is checked and confirmed if applicable to customer • TotalOwed is not in delinquent period • UserKey is updated with new session info
PostConditions	• FreeSpotKey's total spots are reduced by one • Elevator Returns to ground floor if applicable. • Both elevator doors are closed to prevent access to the garage.

Operation	Exit
PreConditions	• FreeSpotKey is updated to show spot availability one the car leaves it's spot • License plate is read by the exit license plate reader and sent to the PaymentController by the LicensePlateOperator. • PaymentController sends license plate information to UserChecker to identify the user to determine payment information in the UserKey. • PaymentController sends payment request to the PaymentOperator to obtain payment if applicable

	• PaymentController confirms payments have been appropriated then sends message to GateOperator to open exit gate. • PaymentController sends message to update the UserKey of payments and the time the session has ended through the PaymentChecker
PostConditions	• GateOperator is instructed to close gate by the PaymentController once the car has exited the garage

C. Mapping Domain Concepts to Use Cases

UC-1: the User selects the register prompt on the InterfacePage. This is sent to the WebPageController to instruct the PageMaker to gather information that is needed to instantiate a new customer such as the billing info, username, and password. This prompt information is gathered and returned as a page to the InterfacePage. The User then enters his information into the prompts to register with AUTOmatedPARK.

UC-2: the User enters their username and password into the prompt on InterfacePage to login to their account. This information is then received through the WebPageController and submitted to the PageMaker to reference the database as to the identity of the user. After a match is found, the information in a user’s profile is sent to the InterfaceMaker to display to the User. The user then inputs that they wish to make a reservation to the InterfacePage. This request is transmitted through the aforementioned channel to return the information on spot availability to the InterfacePage. These user inputs his choice and an optional LicenseNumber that he it is returned to the database as a reservation.

UC-3: As the user pulls into the lift, his plate is read by the LicensePlateReader if the license is not registered he can enter his Username and Password into the KeyPad. The KeyPadOperator or the LicencePlateReaderOperator would then transmit this information to the ElevatorController where it would then be sent to the UserChecker to request a database lookup based on the input, from the UserKey. The identity is then taken to the ElevatorController by the UserChecker. The ElevatorController has also sent the request to SpotChecker for parking availability. SpotChecker then requests the info from the FreeSpotKey that synchronizes with the database to return spot availability. This is return to the ElevatorController that then calculates where (or if) the User should park based on the customer type and spot availability. The result is displayed by the LiftMonitor through the LiftMonitorOperator from the

ElevatorController. If the user can be admitted, the monitor would display where the spot and floor would be and then it would use the ElevatorOperator to operate the Elevator so as to deliver the user to the floor.

UC-4: The database runs half/hour checks to inform and overstayed user. If the SensorOperator has not read from the Sensors that a particular spot is occupied, it will not return the event to the database. Therefore, the database will know that an overstay has occurred, based on referencing its current time minus the time in its records for the completion of the reservation. In this event, the email is sent.

UC-5: the User enters their username and password into the prompt on InterfacePage to login to their account. This information is then received through the WebPageController and submitted to the PageMaker to reference the database as to the identity of the user. After a match is found, the information in a user's profile is sent to the InterfaceMaker to display to the User. The user then inputs that they wish to Manage Reservations to the InterfacePage. The request to either modify or cancel their existing reservations is sent to the WebPageController to get the page from the PageMaker after it retrieves the reservation info about the user to be returned to the InterfacePage. The user may now input their request.

UC-6: As a car leaves the spot, the Sensors change of state is received by the SensorOperator and the SensorChecker is used to update the FreeSpotKey that then synchronizes with the database. The car then approaches the LicensePlateReader2 and the string it reads is received by the LicensePlateReaderOperator. The PaymentController then commands the GateOperator to open the Gate.

UC-7: the Admin enters their username and password into the prompt on InterfacePage to login to their account. This information is then received through the WebPageController and submitted to the PageMaker to reference the database as to the identity of the user. After a match is found, the special Admin view is sent to the InterfaceMaker to display to the Admin. He then inputs that he wishes to SetPrice to the InterfacePage. This request is transmitted through the aforementioned channel to return the information on spot availability to the InterfacePage.

8. A. Extra Credit

Customer Type

Z = f(P, Q, R)

Where:

Z = Customer Type; dependent on whether a customer is recognized, registered, and/or reserved.

P = Recognized | P’ = Not Recognized

If the input from camera matches a record in the database, then the customer is recognized. Otherwise, the customer is not recognized.

Q = Registered | Q’ = Not Registered

If the customer’s e-mail and password match that of an account in the database, then the customer is registered. Otherwise, the customer is not registered.

R = Reserved | R’ = Not Reserved

If the customer admits that he/she is reserved, enters log-in details and/or he/she is recognized as having a reservation in the database, then the customer is reserved. Otherwise, the customer is unreserved and is a walk-in.

Input Permutations

Recognized P	Registered Q	Reserved R	Customer Type Z
No	No	No	Walk-in
No	No	Yes	Invalid
No	Yes	No	Walk-in
No	Yes	Yes	Reserved
Yes	No	No	Invalid
Yes	No	Yes	Invalid
Yes	Yes	No	Walk-in
Yes	Yes	yes	Reserved

Note: 1 = Yes, 0 = No

Input Combinations (yes=1, and no=0) mapped onto Karnaugh maps for each type of customer.

Walk-in

	P’		P	
	P’Q’	P’Q	PQ	PQ’
R’	1	1	1	
R				
	Q’	Q		Q’

Walk-in = P’R’ + QR’

Walk-in customers will not be recognized and will not be reserved or they will be registered and unreserved.

Reserved

	P’		P	
	P’Q’	P’Q	PQ	PQ’
R’				
R		1	1	
		Q		

Reserved = QR

Reserved customers will be registered and reserved.

Valid Customer

	P’		P	
	P’Q’	P’Q	PQ	PQ’
R’	1	1	1	
R		1	1	
		Q		

Valid Customer = Q + P’R’

A valid customer is registered or unrecognized and unreserved.

State machine: Verify Customer

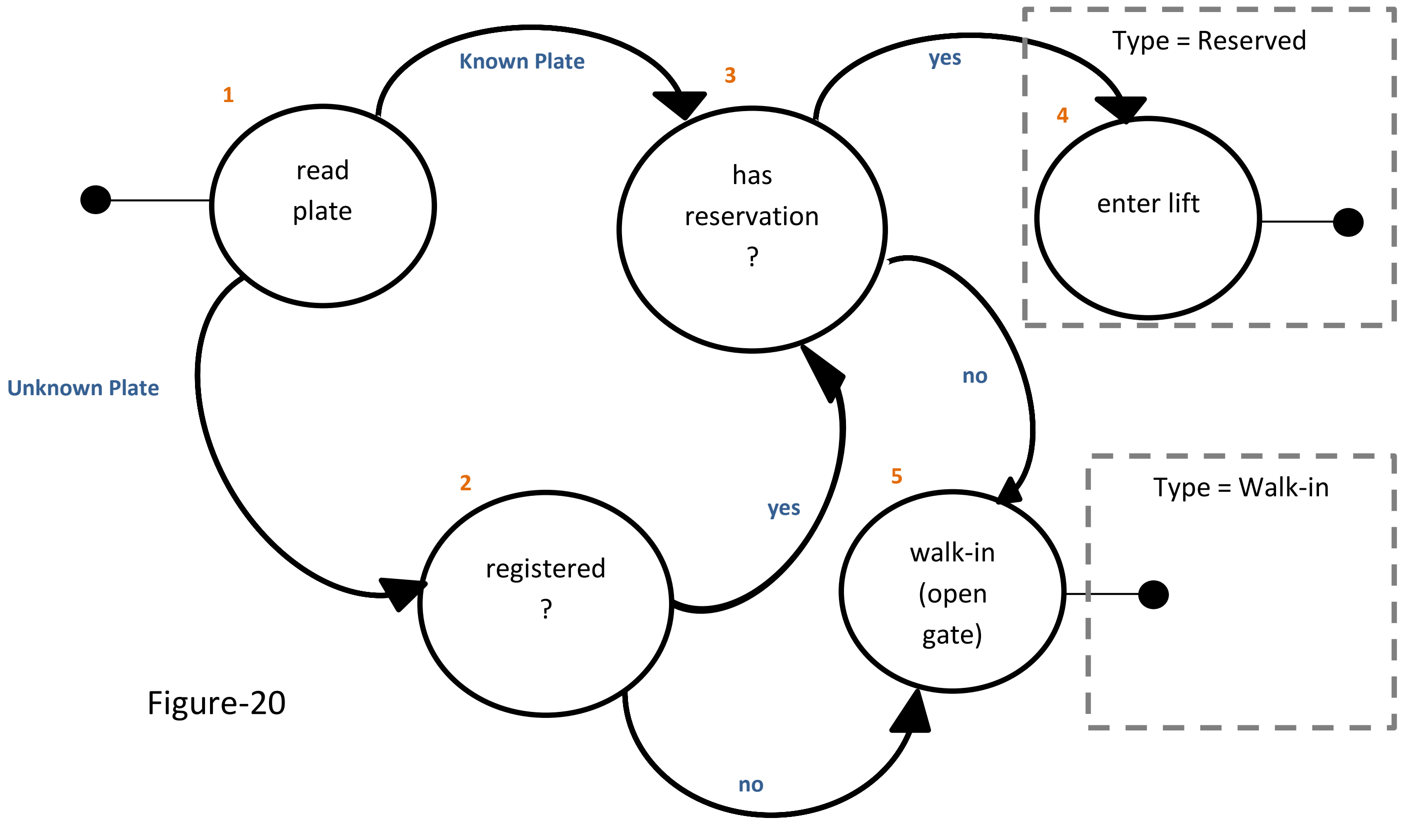


Figure-20

States:

1. Verify Plate (Enter camera + DB)
2. Verify Registration (Console + DB)
3. Verify Reservation (DB)
4. Transport to Floor (lift)
5. Entering Gate (gate)

In the first state, the car's plate number is read. If recognized, the system enters state 3, which checks reservation.

If a customer is reserved, then the system enters state 4 and the car enters lift. If the customer is unreserved, the customer is a walk-in--the system enters state 5 and the gate to the ground floor will open to allow the customer entrance.

If the plate number is unrecognized, the system enters state 2 and checks registration. If the customer is registered, the system enters state 3 and checks reservation. If the customer is not registered, the system enters state 5 and declares the customer as a walk-in by opening the gate to the lowest floor.

8. B. Interaction Diagrams

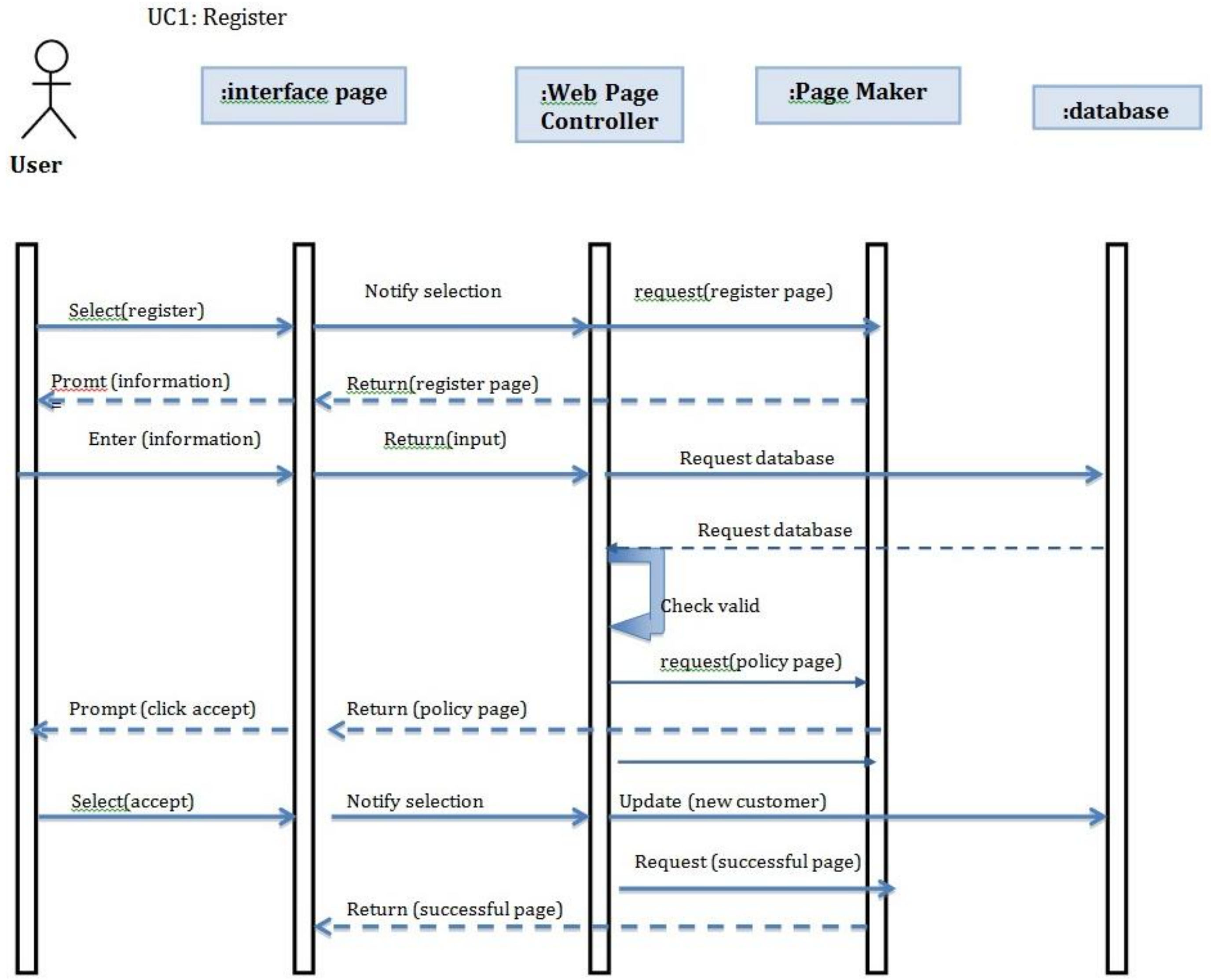


Figure-21

Process:

Customer who wants to make new account clicks on Register on the website of the system, :Interface Page receives the information and send it to :WebpageController. Webpagecontroller requests a “register page” from Pagemaker. Pagemaker sends back a web page to :Interface page and display it user. User has to fill in the information required. After that, information is checked if valid or not. If the information is valid, a new page including the policies of the garage will be displayed. User is prompted to click accept then a successful page will be display to show that user have registered successfully.

Design Principles:

This use case follows state design pattern. In this case, the behavior of WebpageController depends on its state in a complex way. So here, using state pattern is suitable.

UC-2: Reserve

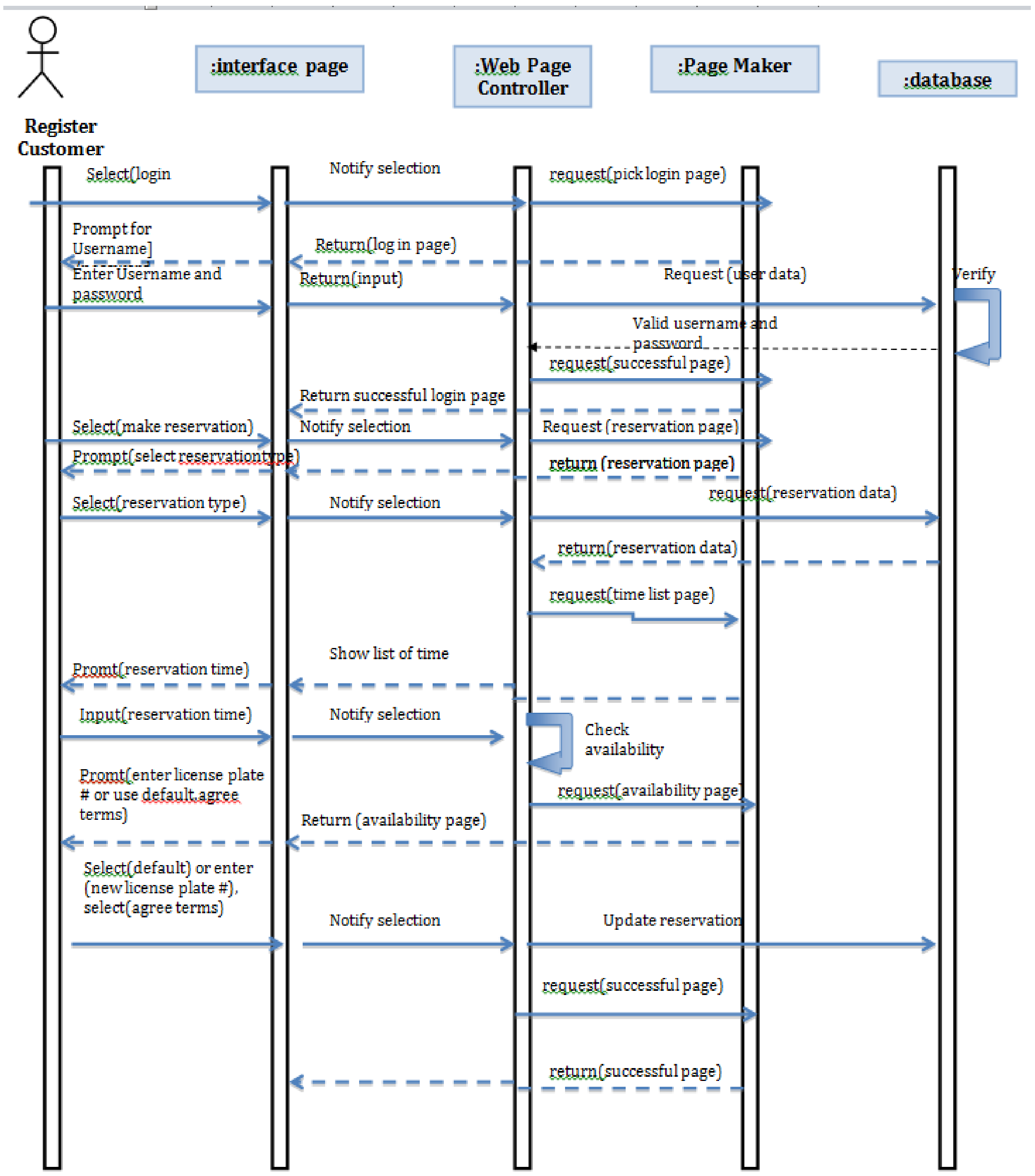


Figure-22

Process:

User clicks on the login. Controller requests the page maker for the login page. A login page is returned to the interface page. User is prompted to enter the password and username. These information will be compared with the database to check if the password and user name is valid or not. A reservation is only made if the user is authenticated. Upon the authentication, the page maker will return a successful page to show that the user have logged in successfully, User clicks on “make reservation”, controller will request page maker to return the reservation page. User is prompted to choose the reservation type, the controller requests the reservation data from server and returns it to controller. Controller again requests Page maker a list of time slots. User is prompted to choose a time slot, and then the controller checks the availability of that time slot. If the time is available, Controller will request Page maker a new page to show that the time slot is available. On new page, users also have to choose to enter a new license plate number or keep the default number, and agree terms of parking. Controller updates the new reservation to the database and requests page maker to return a successful page.

UC-2: Reserve (Alternate Scenario)

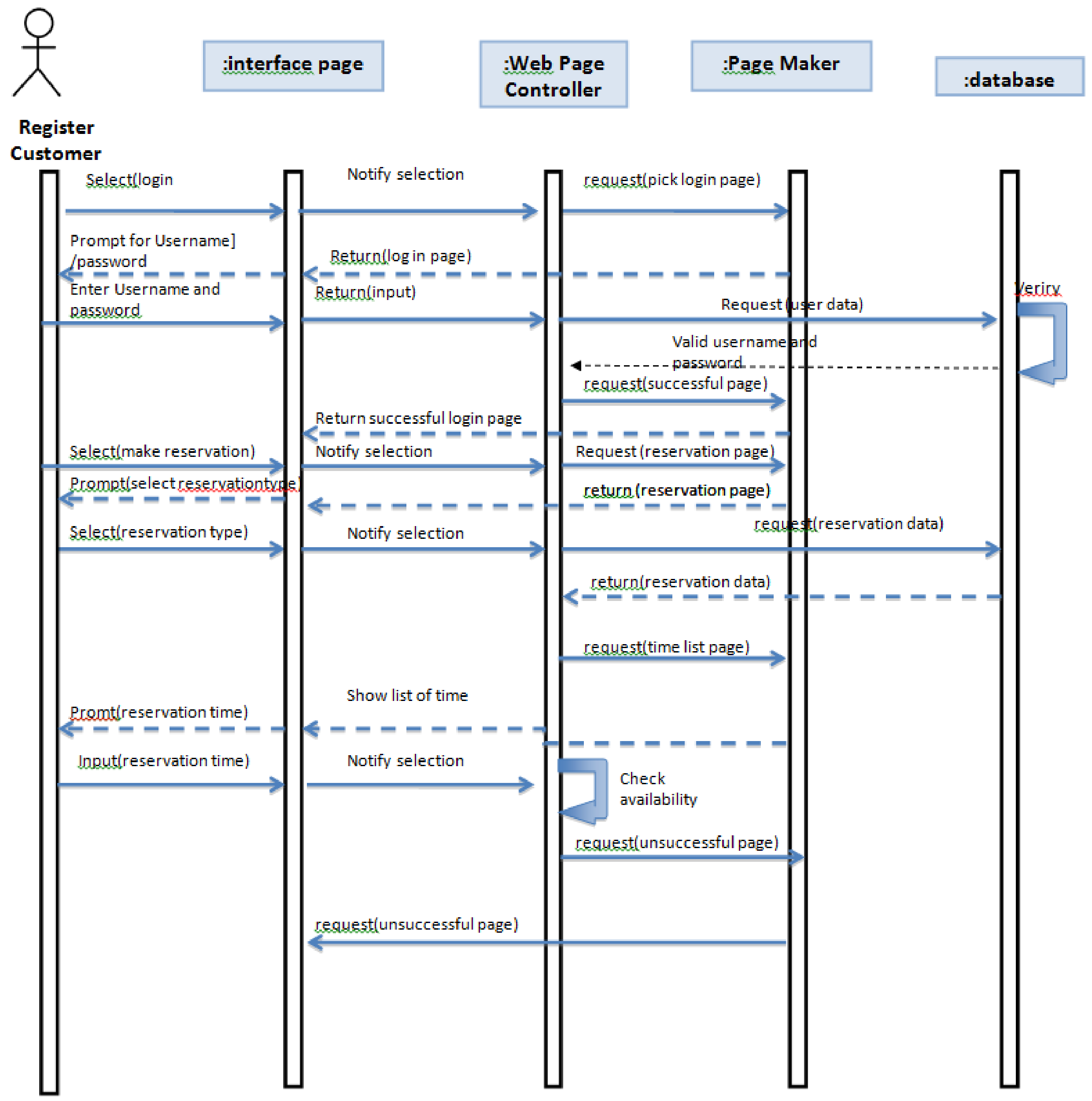


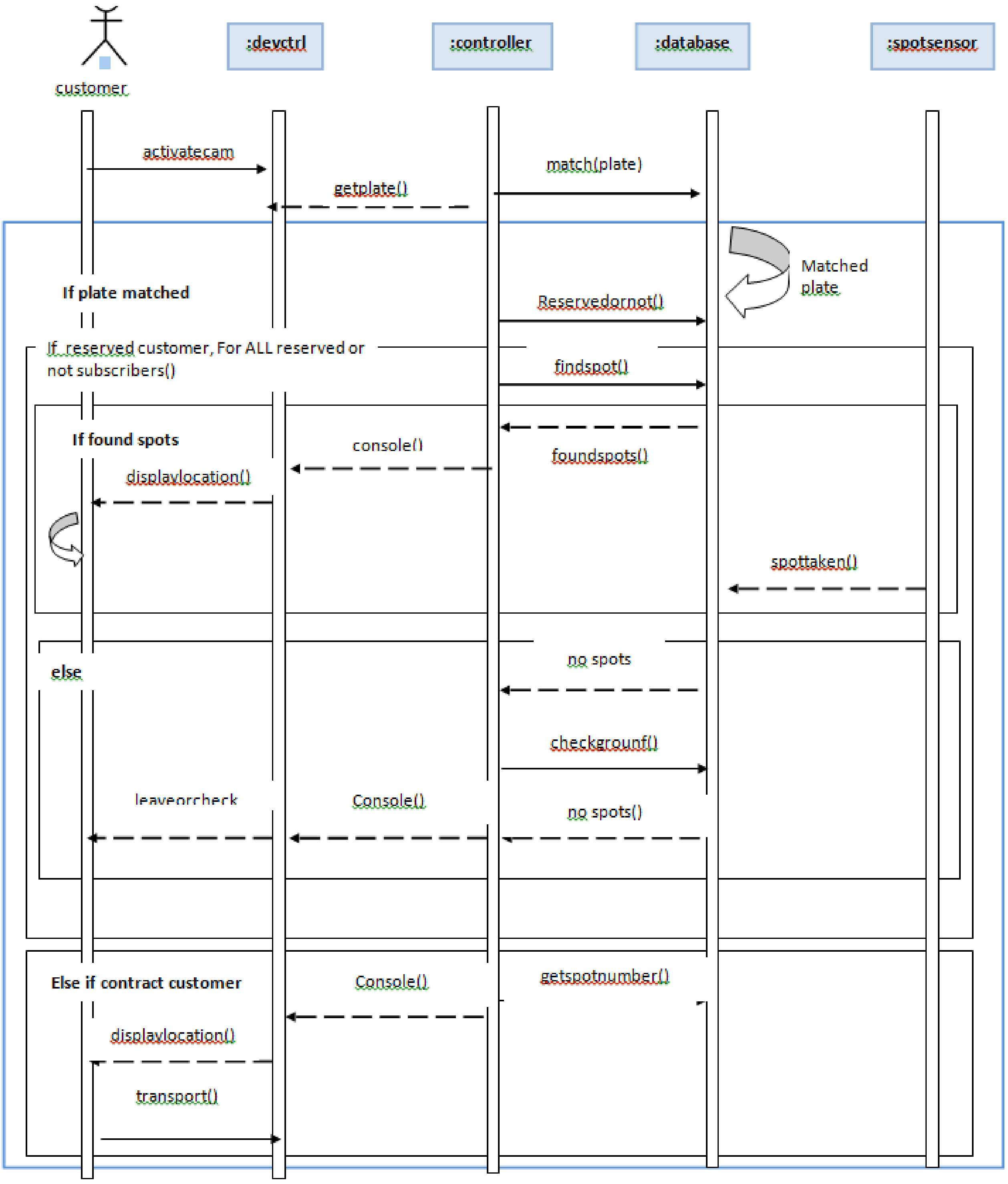
Figure-23

In this scenario, there is no available spot for the time slot that the user has chosen, the controller requests Page make to return a new pageIn the new page, user is advised to check back later because spots can become available due to possible cancellation of reservations.

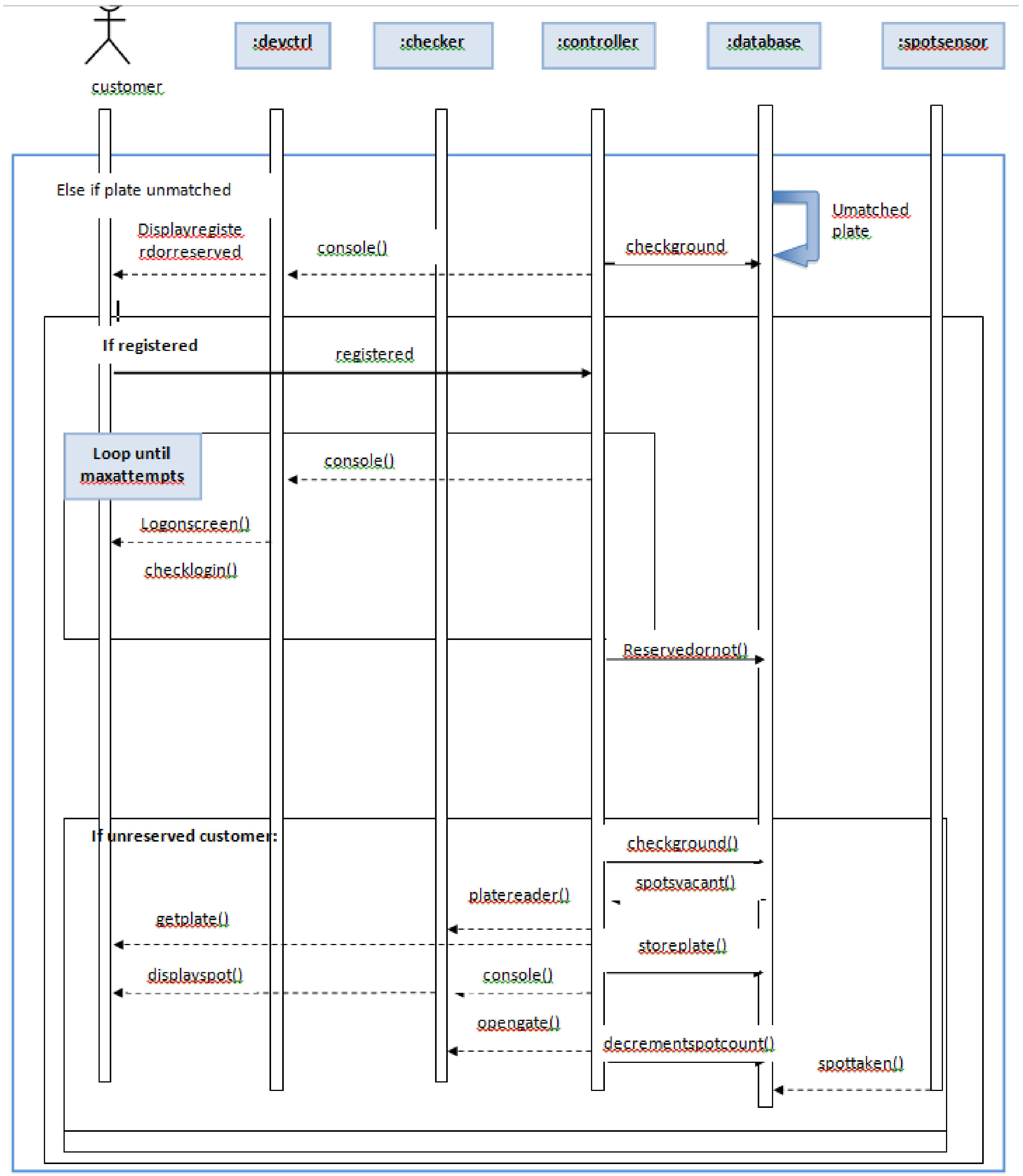
Design Principles:

This use case follows state design pattern. In this case, the behavior of WebpageController depends on its state in a complex way. So here, using state pattern is suitable.

UC-3: EnterLift



Continued… Figure-24



Continued...

Figure-25

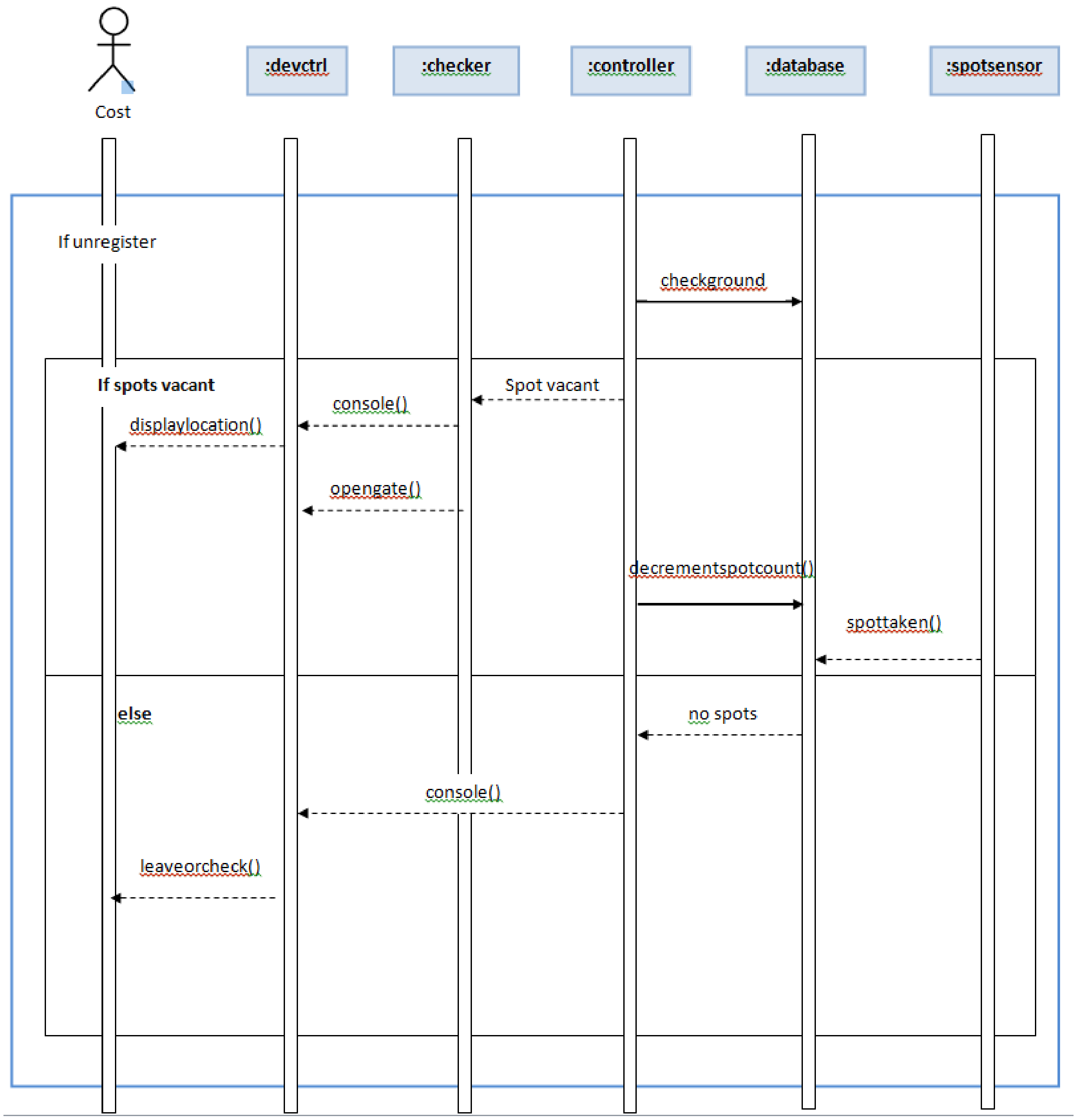


Figure-26

Once a customer enters the plate reader reads the plate and identifies whether the number is recognized or not. If recognized and reserved, then the customer is transported to their desired floor. If there are no spots, check ground floor. If ground is vacant, the customer will be allowed to park in ground. Otherwise he will be asked to leave or accept a rain check. If the customer is a contract customer, the customer will be notified of the spot number and taken to the appropriate floor without updating the database.

If the customer is unrecognized, then the customer is asked to choose between registered or reserved. If registered, he/she is asked for login details. *The reservedornot function is called and the reserved customer then becomes subscriber to this publisher*. The unreserved customer is lead to the ground floor and linked to his/her account. The unregistered customer is a walk-in who will be led to the ground floor if there are spots and he/she will be asked to leave if there are no available spots.

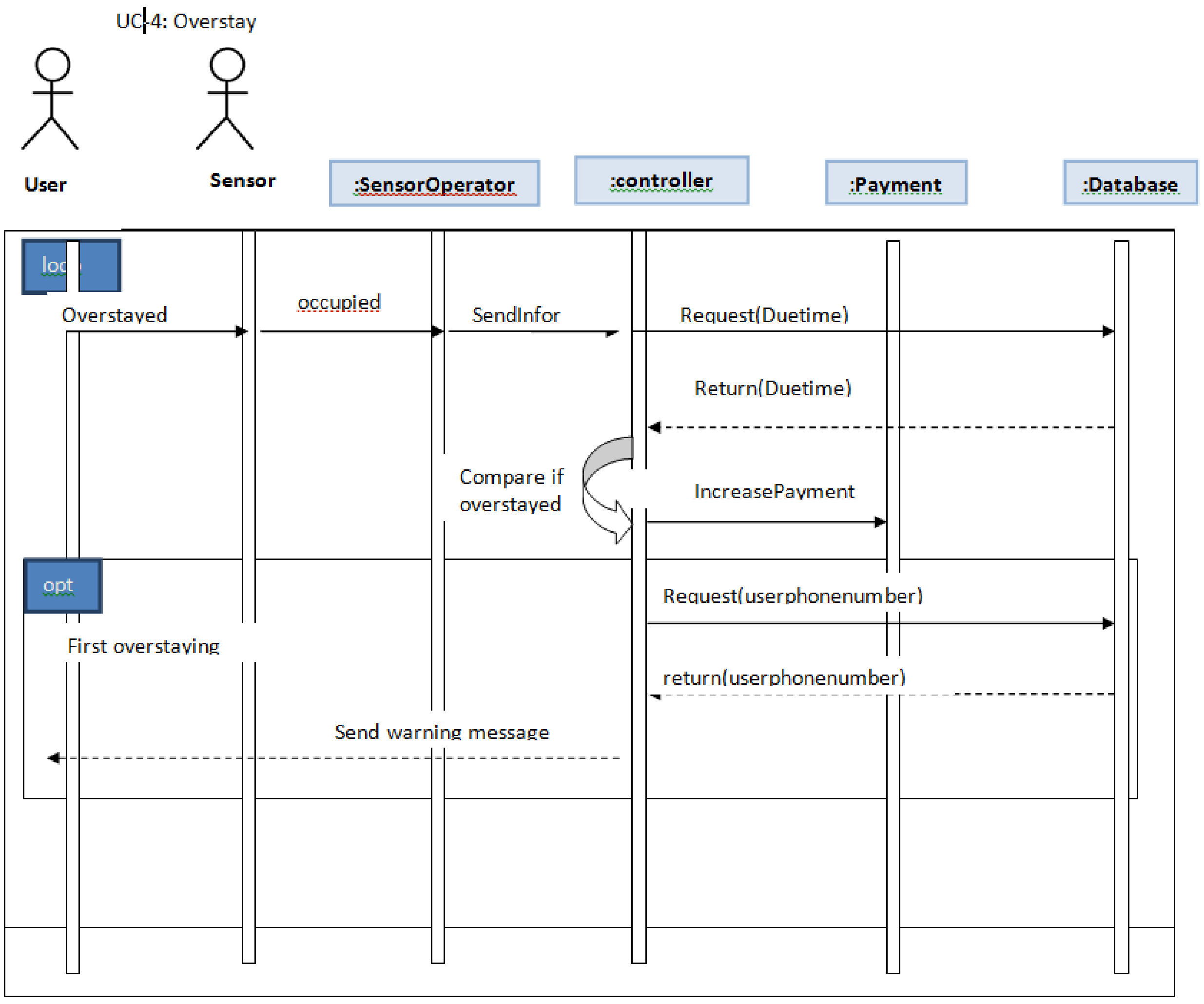


Figure-27

Description: When customers are parking, the sensor will check the occupation of the car in every 30 minutes (admin can change this period of time). Then the sensor will send staying time to controller. Controller will send request to database to take the due time of the car, then compare with the current time. If the car is determined to overstay, the payment of the User will increase at a specified rate. In the first time of overstaying, controller will request the phone number of customer and then will send a message to customer to warn them about their overstaying.

UC – 5 User Manage

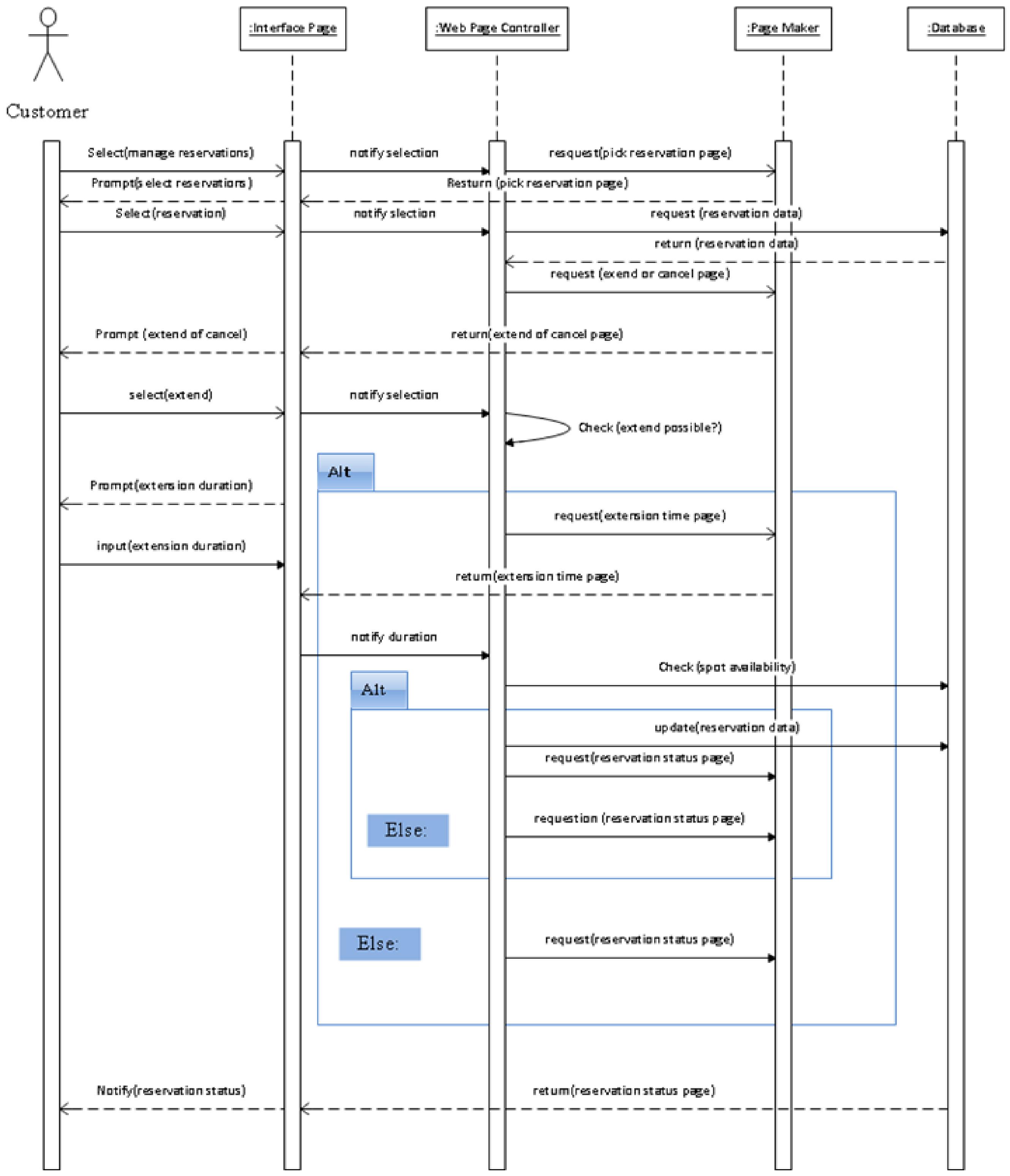

Customer
:Interface Page
:Web Page Controller
:Page Maker
:Database
Select(manage reservations)
notify selection
request(pick reservation page)
Prompt(select reservations)
Resturn (pick reservation page)
Select(reservation)
notify slection
request (reservation data)
return (reservation data)
request (exend or cancel page)
Prompt (extend of cancel)
return(extend of cancel page)
select(extend)
notify selection
Check (extend possible?)
Alt
Prompt(extension duration)
request(extension time page)
input(extension duration)
return(extension time page)
notify duration
Check (spot availability)
Alt
update(reservation data)
request(reservation status page)
Else:
requestion (reservation status page)
Else:
request(reservation status page)
Notify(reservation status)
return(reservation status page)

Figure-28

Alternate scenario (During Registration):

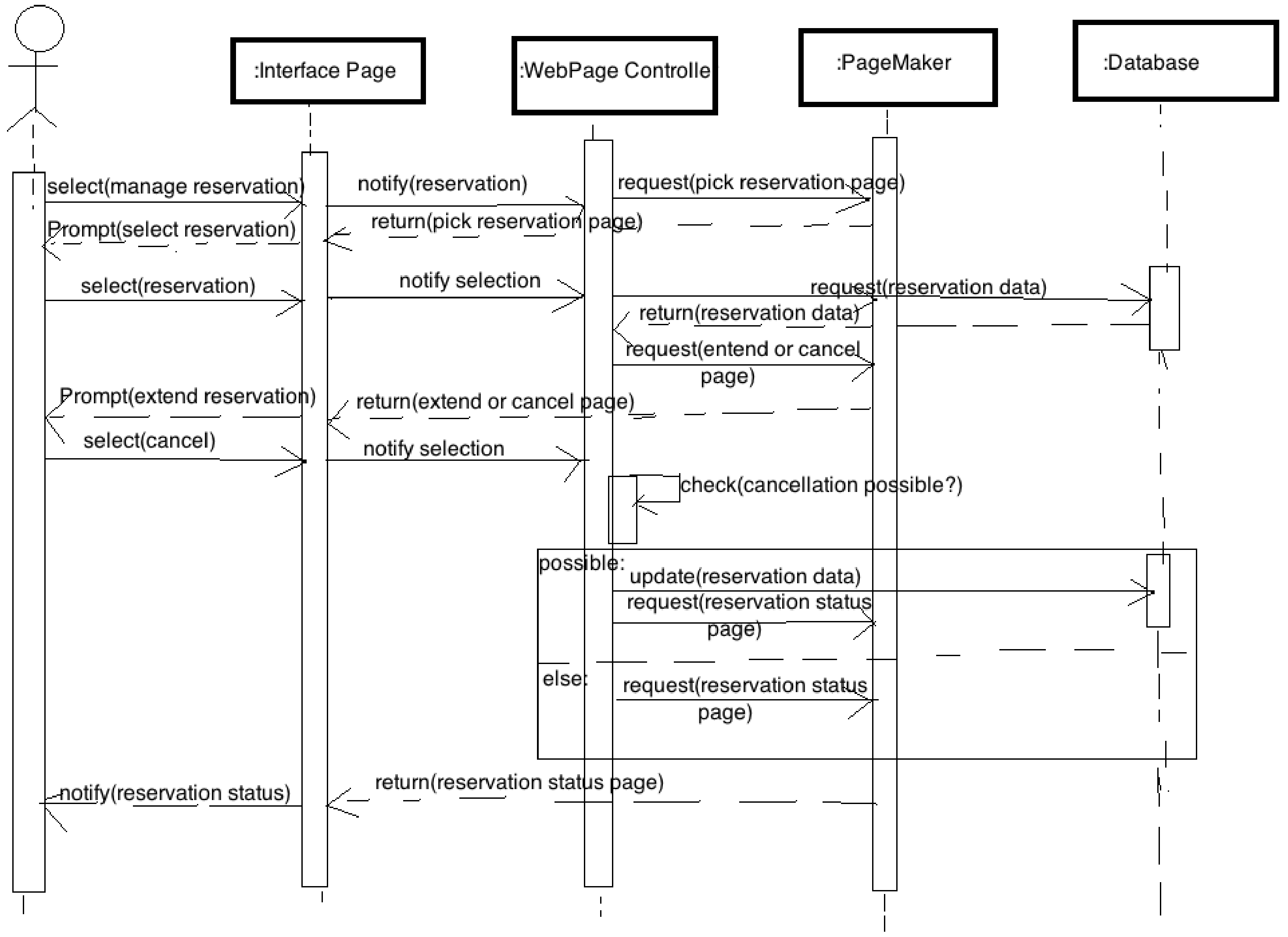

:Interface Page
:WebPage Controlle
:PageMaker
:Database
select(manage reservation)
notify(reservation)
request(pick reservation page)
return(pick reservation page)
Prompt(select reservation)
select(reservation)
notify selection
request(reservation data)
return(reservation data)
request(entend or cancel page)
Prompt(extend reservation)
return(extend or cancel page)
select(cancel)
notify selection
check(cancellation possible?)
possible:
update(reservation data)
request(reservation status page)
else:
request(reservation status page)
return(reservation status page)
notify(reservation status)

Figure-29

UC-5 Design Principles Implemented:

The configuration of this use case follows that of a state pattern. The state of attributes of the context objects (the user and the database) are received by the interface page and PageMaker, respectively. The state is then transmitted to the WebPageController that responds dependent on the information it receives from the context object. It is relatively basic and straightforward implementation of a GUI so it was a benefit.

UC-6: Exit

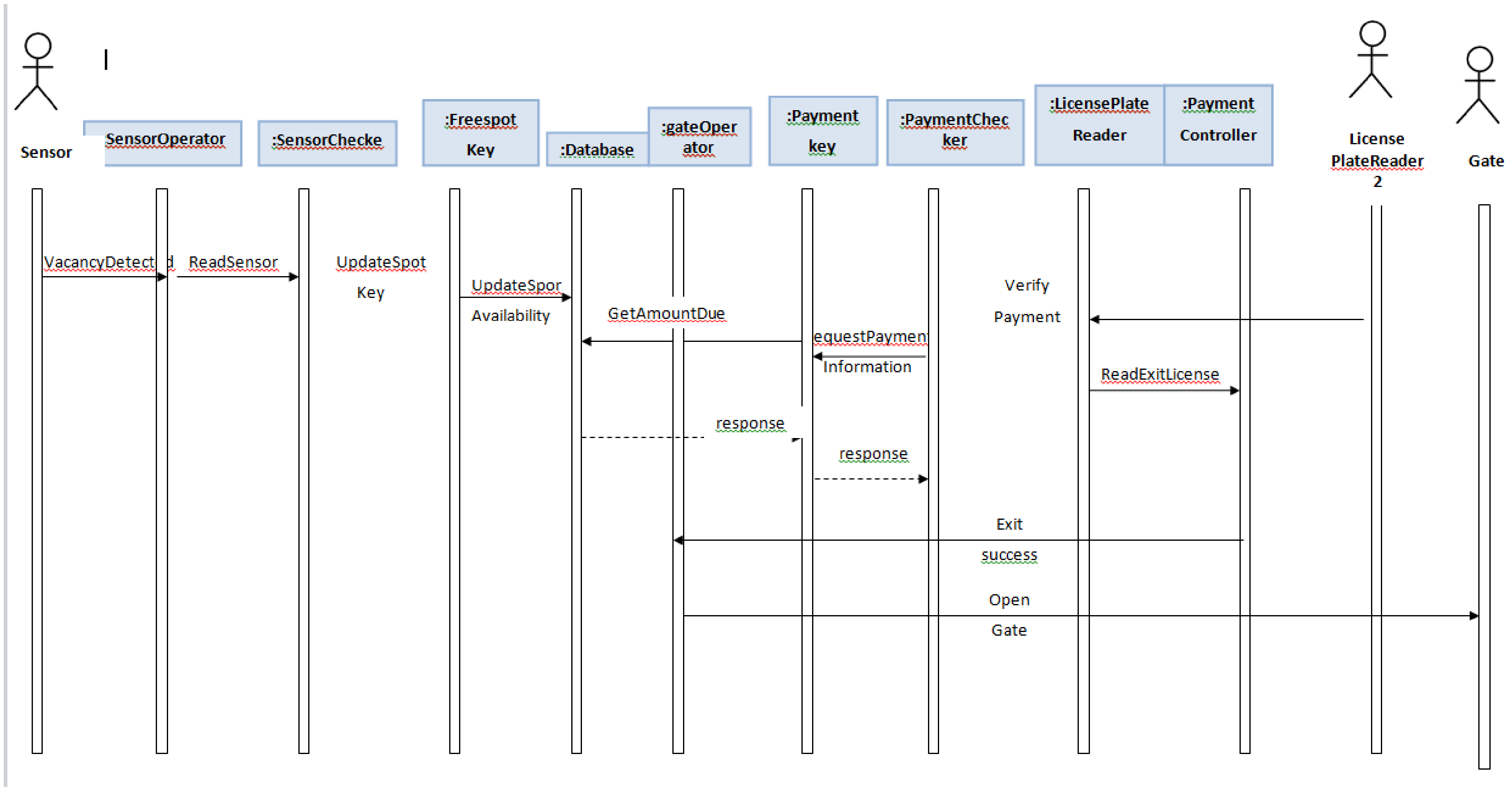


Figure-30

UC-7: SetPrice

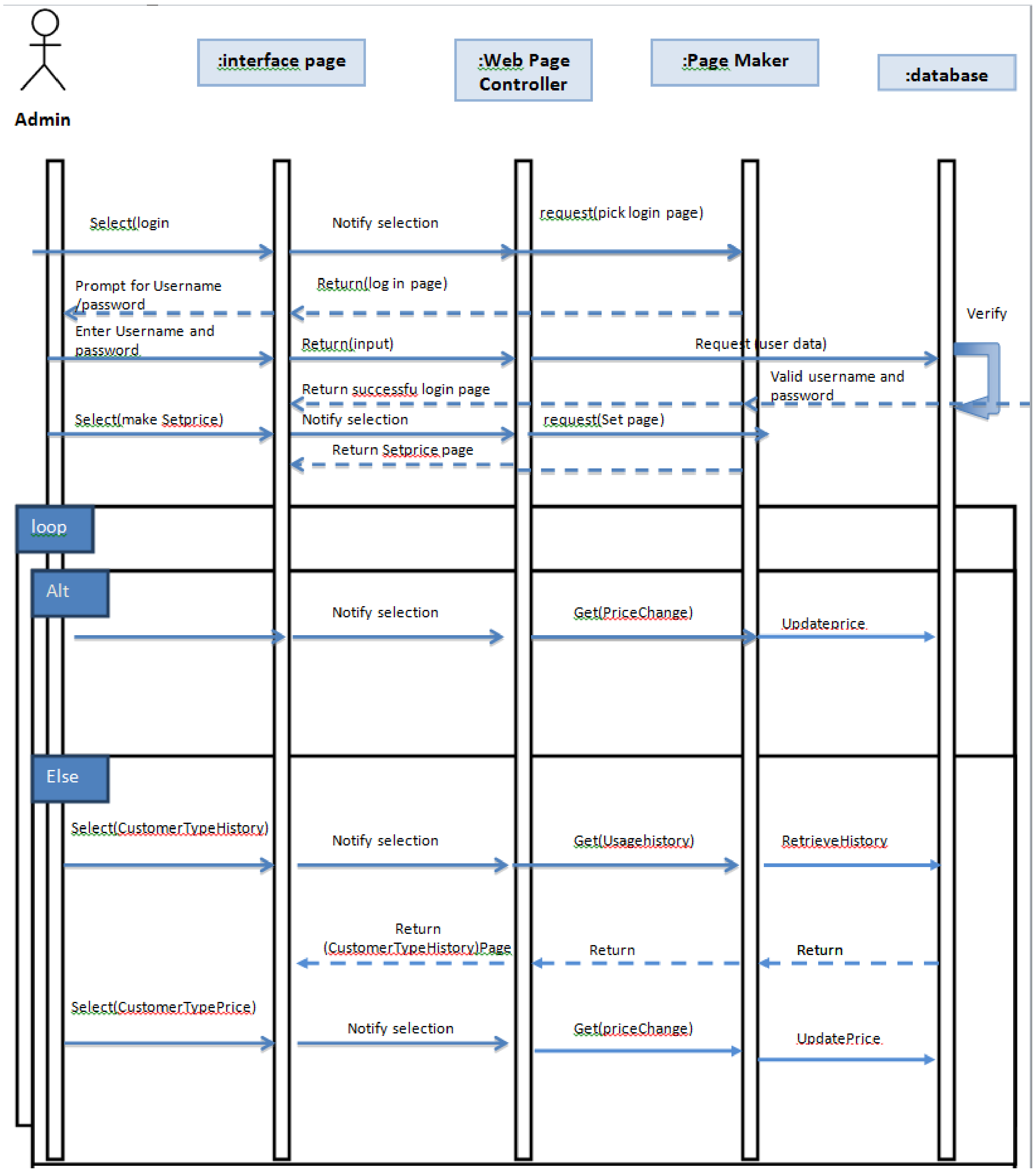


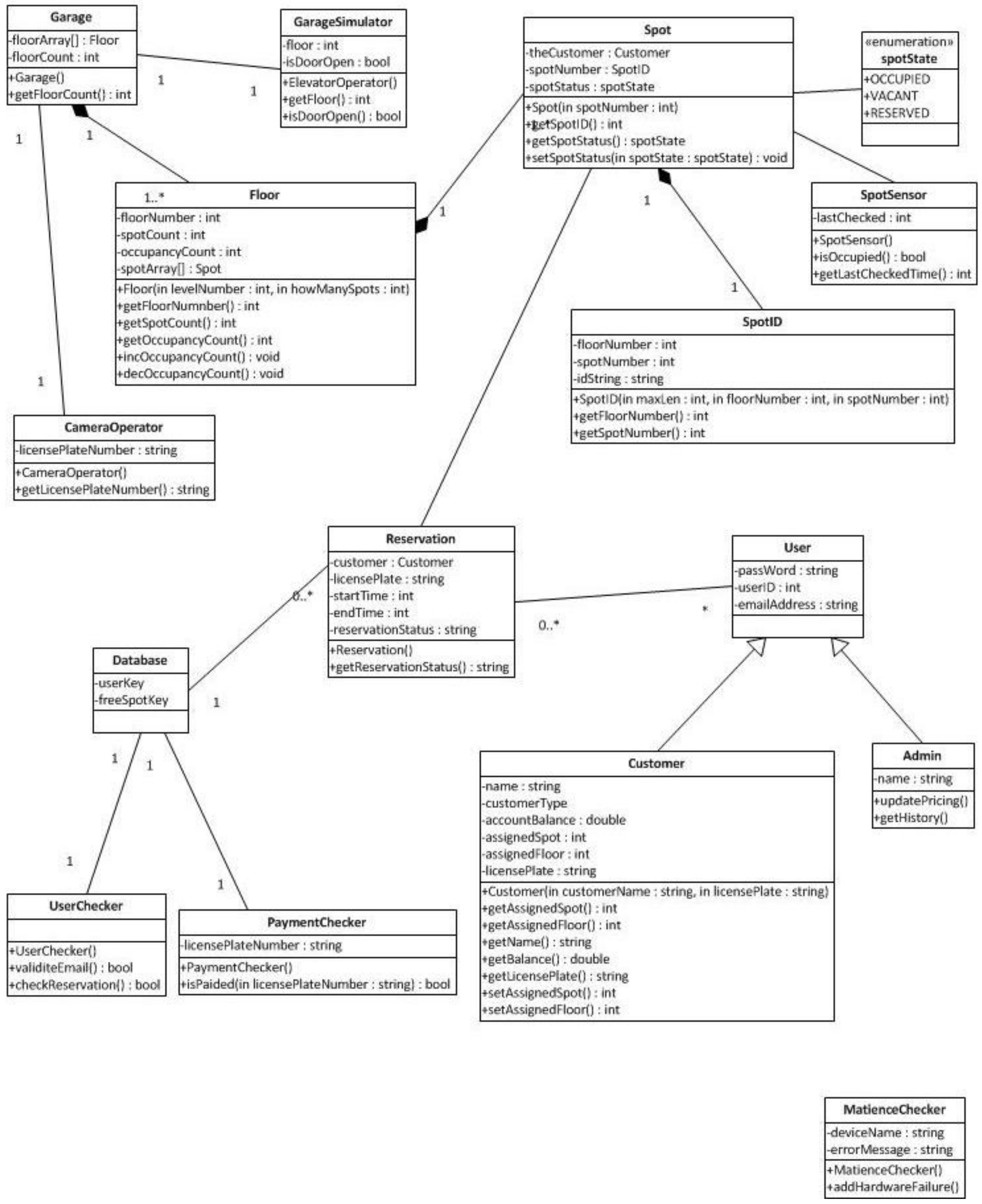
Figure-31

UC-7 Design Principles Implemented: Much like the UC-5 Design Principals used, this utilizes the State Pattern for the same reasons

9. Class Diagram and Interface Specification

A. Class Diagram

Figure-32



ElevatorController has been renamed to GarageSimulator.

B. Traceability Matrix

	Garage	Spot	SpotID	SpotState	SpotOperator	Reservation	Floor	User	Admin	Customer	UserChecker	ElevatorOperator	CameraOperator	PaymentChecker	UserKey	FreeSpotKey	MaintanenceLogger
SensorOperator		X	X	X	X												
SensorChecker		X	X	X	X		X										
ElevatorOperator												X					
GarageSimulator												X					
LIcenseOperators													X		X		
UserKey						X											
PaymentChecker														X			
UserChecker						X		X			X						
KeypadOperator												X					
FreeSpot Key		X	X	X	X		X									X	
DoorOperator																	
MaintanenceController																	X
MaintanenceKey																	X

	Garage	Spot	SpotID	SpotState	SpotOperator	Reservation	Floor	User	Admin	Customer	UserChecker	ElevatorOperator	CameraOperator	PaymentChecker	UserKey	FreeSpotKey	MaintenanceLogger
GateOperator																	
PaymentController														X			
Payment Operator														X			

The classes depicted in the class diagram and detailed in the Operation Signatures have evolved from key domain concepts. While a few concepts were found unnecessary, other's names were modified when they were applied to classes, and other yet became abstract classes, based off of which other classes will be derived.

A key important concept was the UserChecker. The UserChecker takes in the license plate number from the Camera and pulls up any user record associated with that plate. It gives details about any reservations that the user has made to the GarageSimulator which then would use this information to display the appropriate messages in the Elevator console as the user entered the elevator.

We have retained the name UserChecker and made this a class with no attributes. It merely hosts two functions that check if the license plate number is in the database, and if this is the case, pulls up customer related records. For this, a class Customer has been created, which would be the data-type returned by lookUpCustomerInfo() called by UserChecker.

In the domain model, it is the GarageSimulator that calls UserChecker's functions. But GarageSimulator has not been included as a class. GarageSimulator is a mere delegator of tasks and does not do have any functions or attributes of its own. It merely coordinates the actions of the overall setup. To do this, we would have the main function itself coordinate these calls instead of creating a separate GarageSimulator class that cannot be instantiated. GarageSimulator deals with the control flow of the system, and our MAIN function will coordinate this control flow.

In the domain model, we had a SpotOperator and a SpotChecker. On deriving related classes, these were replaced by Spot, SpotID, SpotState, SpotOperator. The Spot class has a Spot ID and a SpotState (occupied vs reserved vs vacant). The SpotOperator updates information whenever a spot is occupied.

The PaymentChecker and PaymentController were another important aspect of the domain model. Here, PaymentChecker is retained as a class. PaymentController has no function call or attributes of its own but merely controls the flow of execution of the program. This flow will be incorporated in the Control and not created as a separate class.

The MaintanceLogger class also takes from the maintenance concepts of the domain model. It logs any device failure and the error message, then later sends it to a Maintenance Manager of the garage.

C. Design Patterns

The pub-sub pattern is being used in this design. The subscriber does not care for the publisher. This was done to hide implementation details from outside classes. This helps provide modularity, which is a key principle of object-oriented programming. This helped us code by allowing us to use the publishers repeatedly and not have to code the same thing each time. One block of code could be used multiple times in multiple ways, which reduced the complexity of the program. Thus the use of the publisher-subscriber design pattern has helped us to prevent writing the same lines of code repeatedly.

D. Object Constraint Language (OCL) Contracts

Class Garage

Invariant: Has floors

Context Controller **inv:**

Self.getFloorCount > 0

Pre-condition:

Post-condition

____

Class Floor

Invariant: Has Spots

Context Controller **inv:**

Self.getSpotCount > 0

Pre-condition:

Post-condition

____

Class Spot

Invariant: has spotID

Context Controller **inv:**

Self.getSpotID ~= NULL

Pre-condition: No customer occupying spot.

Context Controller: spotState **Pre:**

Self.getSpotStatus == VACANT;

Post-condition: Customer occupies spot.

post: Self.getSpotStatus == OCCUPIED || RERSERVED;

____

Class User

Invariant: User with a valid email and password

Context Controller **inv:**

email && password == valid;

Pre-condition: Must be registered user or admin.

Context Controller: string **pre:**

If customerType == 'REGISTERED' || User == admin

Then proceed;

Post-condition: Adds reservation information.

Post: getAssignedSpot ==valid && getAssignedFloor == valid

____

Class Reservation

Invariant: User with a valid email and password

Context Controller **inv:**

email && password == valid;

Pre-condition: garage must be available.

Context Controller: string **pre:**

If reservationStatus == free

Then proceed;

Post-condition: Decrement in total spot count.

Post: garage**.**floor.spotcount = garage**.**floor.spotcount -1;

10. System Architecture and System Design

A. Architectural Styles

The IEEE defines Software Architecture as the "fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution."[1] Though many problems in software engineering can be considered unique, system requirements usually share broad commonalities such as communication, data storage and security/reliability. Software Architecture seeks to provide for basic blueprints with which to resolve these issues. Several fundamental organizations or "styles" of architecture have arisen that are consistently helpful when addressing particular problem domains. As is often the case, the complexities of our requirements caused us to incorporate several such styles in varying degrees.

The Pipe-And-Filter model is a good representation of our method for obtaining user information as they pull up to enter or exit the garage. Examples of the "pumps" or producers of our data are the License Plate Readers, Key Pads, and Parking Sensors. The data is then "filtered" through a series of Operators which are drivers for the hardware components. This data is then collected by our Checkers which interacts with our Keys to determine the use case and then presents that information to the Controllers to calculate a response. The "sink" is the database which will ultimately receive the input user information as well as the system response to that input.

State Driven Architecture is a structure used to tailor system responses to various events. As the Pipe-And-Filter is bringing information into our System, we use state driven architecture to direct it to the appropriate pipes and filters. Our framework of Parking Sensor, License Plate Readers, Key Pads, and Web Pages, take in information about the state our User is in relative to our System. For instance, as the Elevator Controller receives a license plate number from License Plate Reader 1, the system responds by looking up who the User is to from the Database so that it knows how to process the approaching customer. In essence, the system has shown that it realizes that the User is in the initial parking process state and responds accordingly until it can assist the User into a spot. As the spot is entered, the sensors are tripped and the system registers the User as being in a parked state. Another example is when a User approaches during a time of no occupancy, as the state of the garage doesn't allow for another customer and so they are not be admitted by the ElevatorController.

We use a derivative of the Client-Server model called 3-Tier Architecture to provide for the Garage Operator and Customers to remotely access (or in the case of a new user, create) their profiles. The front-end that our clients use to view their profiles or garage operators use to access their admin pages is called the presentation tier. It is comprised of the HTML pages that are accessed with a web browser. The application tier represents the back-end which processes all requests. It performs all the specific functionality that actually pertains to the web application. It does not, however, store any user specific information. All data storage is left to the 3rd and final tier of the architecture, the database, since it contains the main data management system.

Lastly, we incorporate all of these architectures and system components as a Model-View-Controller for smooth operation. The Model is the Business Logic that we use to govern

[1] IEEE Computer Society, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems: IEEE Std 1472000. 2000.

the way the system responds to the data that it receives. The View is the way that we present the data our Monitors and Gui's display to our Users and Admins. Our Controllers are used to determine the course of action for the system based on the information that we collect from our Users and Admins.

B. Identifying Subsystems

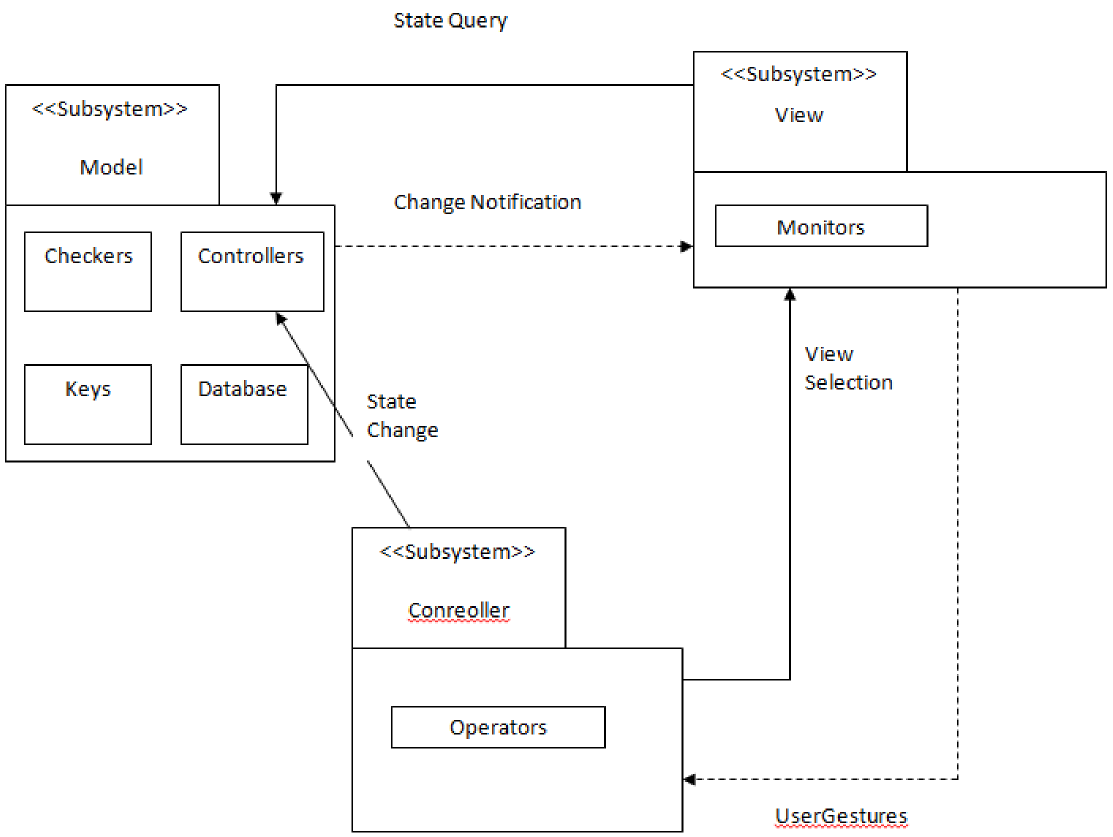


Figure-33

The Package Diagram was implemented in this way because our system is structured as the relationship between Model, View and Controller Subclasses.

The Model Subsystem is comprised of our Business Logic, our stored data and the methods of accessing and updating that data. Our Business Logic is programmed into our controllers (PaymentController, Elevator Controller) and aided through checkers that access our data through our keys and our database.

The View Subsystem renders the contents of a model. It accesses our data through the model and specifies how the data is to be presented. Our Monitors and Guis comprise this subsystem. The controller translates interactions with the view into actions that are to be performed by the model. The actions include activating

The Controller Subsystem manages data received from Actors. The Operators in our system comprise this package. They deliver their input into the Controllers of our Model System so that we may address the situation appropriately.

C. Mapping Subsystems to Hardware

The following are the subsystem that we have in our system

Client Side

The client side is simply the user interface that is created with html and CSS. The user can input information into the webpage forms which will be stored in the database and associated with the user. The user will also be given the privilege to manage reservations, user information, etc. And all the web pages will be displayed to the user to allow for modification of information.

The client would need an internet-enabled device to reserve.

The client side will contain the view subsystem (Monitors) and the controller subsystem (controller). These subsystem fall under the client side because these subsystems are used by the client to trigger the system.

Server Side

The server maintains almost all code and executes processes as needed. The server houses all controllers

The server maintains a database which stores information about the user, user parking statistics for payment purposes, reservations and the parking garage statistics. The user's email, password, account information, address, etc. are all stored in this database. The user parking statistics include the duration of stay, overstays, no-shows, etc. Reservation information includes the date and time of reservation, spot availability, license plate information, etc. Parking garage statistics are for the purposes of the administrator to view logs of incoming and outgoing customers.

The server side will contain the model subsystem which basically contains Checkers, Controllers, Keys and the database. The fall in the server side because this is used by the software to make the program functioning and these do not directly interact with the users.

D. Persistent Data Storage

We must utilize persistent data storage since information generated from a given process must be referenced in order to properly identify a future use case when it is encountered. An example would be a user's profile data must be persistently available so that the User Checker can present any profile information to the Elevator Controller so that the customer can be processed dependent on their type and whether or not they have a reservation or a contract.

For our Database management system, we will use relationship database SQL to manage the database. The version of SQL that we use is SQL 2008. The information that we need to keep track are:

- Customer Profile Information
    - Reservations/Contracts
    - License Plate Character Strings
    - Usage History
    - Billing/Payment info
    - If the customer is in good standing
    - Pass-codes
- Spot Availability
- Maintenance Issues
- Administrators and their privileges

The following is the list of database tables that were used to implement persistent data storage:

<u>Member table:</u>

The following is the list of information that is stored in this table as columns.

firstName

lastName

middleInitial

dateOfBirth

plateNo

cardNo

<u>MemberAddress table:</u>

The following is the list of information that is stored in this table as columns.

streetAddress1

streetAddress2

city

state

zipcode

<u>MemberLogin table</u>:

The following is the list of information that is stored in this table as columns.

userName

userPassword

memberid (auto generated)

<u>OnlineReservation table:</u>

The following is the list of information that is stored in this table as columns.

reservationId (auto generated)

carPlateNumber

reservationStartDate

reservationEndDate

Payment table:

The following is the list of information that is stored in this table as columns.

billingYear

billingMonth

hoursParked

amountDue

paidFlag

creditCardNumber

E. Network Protocol

We did not have to use separate machines to run the system. We imported the database and all the required code onto a single machine and tested the system using just that one machine. Thus we didn't implement any connections or proxies and thus didn't have to use any communication protocols.

F. Global Control Flow

- Execution order:

Our system is procedure-driven and executes in a linear fashion. For example, in order to make a reservation online, the user has to go to the website then use their username and password to log into their account, choose the reservation button, select day, time, location, and then finally submit it.

- Time dependency:

The system waits for the customer to input details when they are in the elevator, after that period, the system will have to warn the user and ask him to leave if he still doesn't provide any inputs. The timer is used to record when customers arrive and leave. Since we must be highly responsive to our customers and be able to communicate these records to them, we will operate our system with real time references. We also have a huge time dependency because online reservation is also done in real time and we have to promptly respond to the user's needs. A delay in processing can result in customer satisfaction and would prevent us from achieving our goals.

- Concurrency:

Our system uses multiple threads. For the server, we need multiple threads when there are more than one customer log in to system and make/modify/cancel their reservations. We use a lot of sensors so we also need multiple threads because they work at the same time. Threads are used for the plate-reader camera, keyboard elevator controller. We must use mutex locks on a particular parking spot for reservations in order to eliminate double booking a spot. This will protect us from double booking the last spot in a high volume period.

G. Hardware Requirements

- LCD in the elevator: This LCD is used display the information to the customers, the display should be 42 inch at least.

- Keyboard: for customer to enter the information. This is a waterproof keyboard.

- A computer: to process information from the keyboard and display the information to the LCD. The resolution should be 1280x720.

- Sensor: to detect if the spot is vacant or not.

- Plate-reader camera: 10Mb at least to read the plate of the card.

- Database: with 100GB at least to store all the information.

- Hard-drive: At least 50 GB of space. This disk will store only cached information.

- CPU 4GHz processor with 4GB of RAM in order to process the information of the keyboard and the LCP.

- Server: to process the information from the user and the garage system

- Internet connection: to connect the garage system with the garage. The speed should be 2Mbps at least

- Internet connection: to connect the garage system with the garage. The speed should be 2Mbps at least

11. Algorithms and Data Structures

Algorithms:

Our system does not use any algorithms. (We do not implement any math model)

Data Structures:

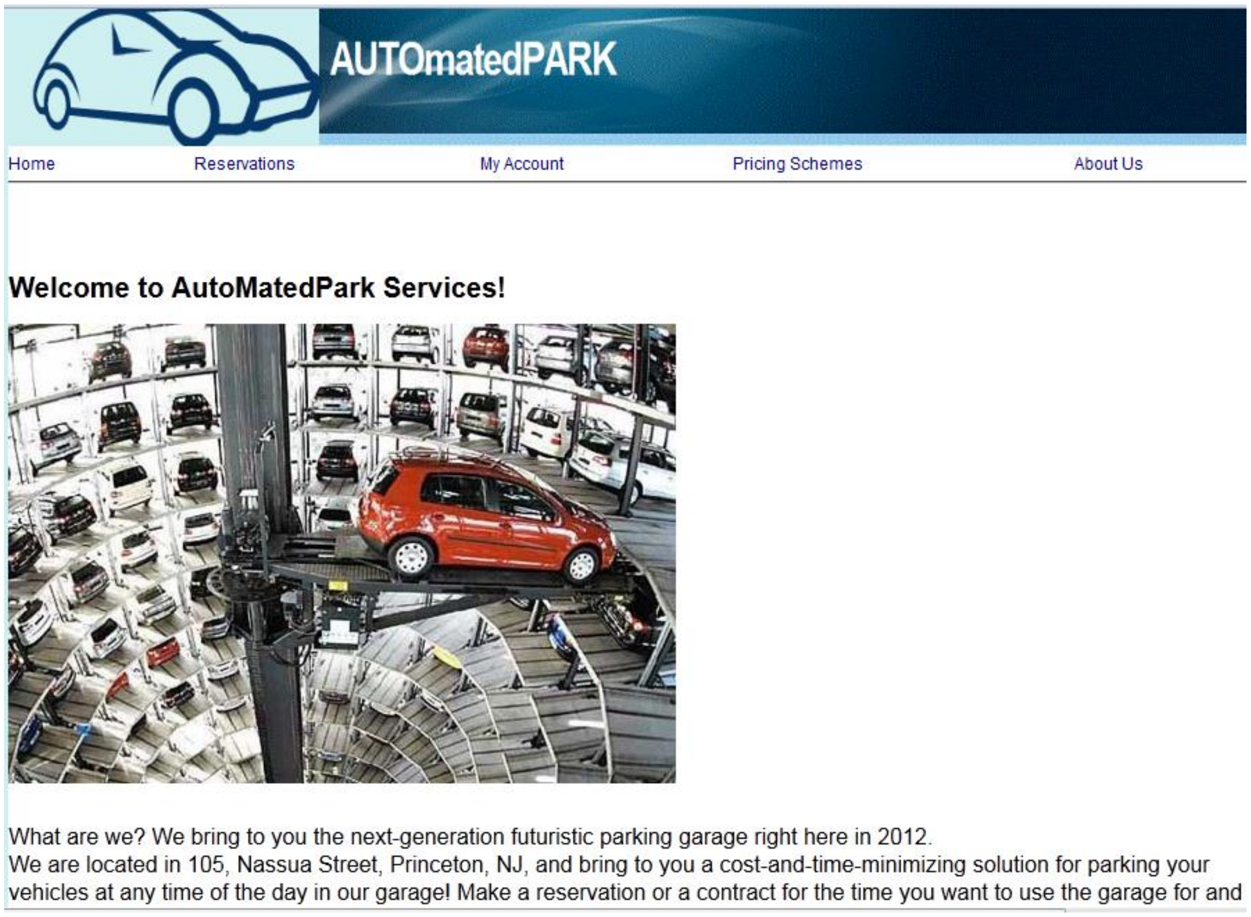
- A color coded array is used to implement the garage display in the admin view. The various occupancy states such as vacant, occupied, overstay, reserved, etc are assigned different colors and represent the status of a spot at a particular time.
- A database table is used to maintain the user login information such as email address and password. The following data types are used to store the variables in the table.
  - Email address: string
  - Password: string
- A database table is used to store member personal information such as first name, last name, middle initial, phone number, credit card number, and license plate number. The following data types are used to store the variables in the table.
  - First name: string
  - Last name: string
  - Middle initial: string
  - Phone number: int
  - Credit card number: int
  - License plate number: string
- A database table is used to store member addresses which contain fields such as house number, street address 1, street address 2, city, state, and zipcode. The following data types are used to store the variables in the table.
  - House number: int
  - Street address: string
  - City: string
  - State: string
  - Zip code: int
- A database table to store and maintain the reservations and contracts. The reservation start time and end time will be stored in this table. The following data types will be used to store information in this table.
  - Reservation start time: datetime
  - Reservation end time: datetime

- A database table to store the user payment details. Data about the billing year, month, hours parked, amount due, paid flag and credit card number will be stored in this table. The following data types are used to store the variables in the table.
    - Billing year: int
    - Billing month: int
    - Hours parked: float
    - Amount due: float
    - Paid flag: Boolean
    - Credit card number: int
- A bit map to maintain and monitor garage occupancy. If the spot is occupied it will be set to one in the bit map and if it is vacant it will be set to zero in the bitmap.
- An array to maintain the list of floors and the number of spots in each floor.
- A temporary array buffer to store the input data during registration or reservation because data will be stored in the database only in the final step. We don’t want to be calling the database very often because this might slow down the process and we’ll have to frequently delete data from the database which is not advisable.

Other than the data structures stated above our system doesn’t use any other complex data structures.

12. User Interface Design and Implementation

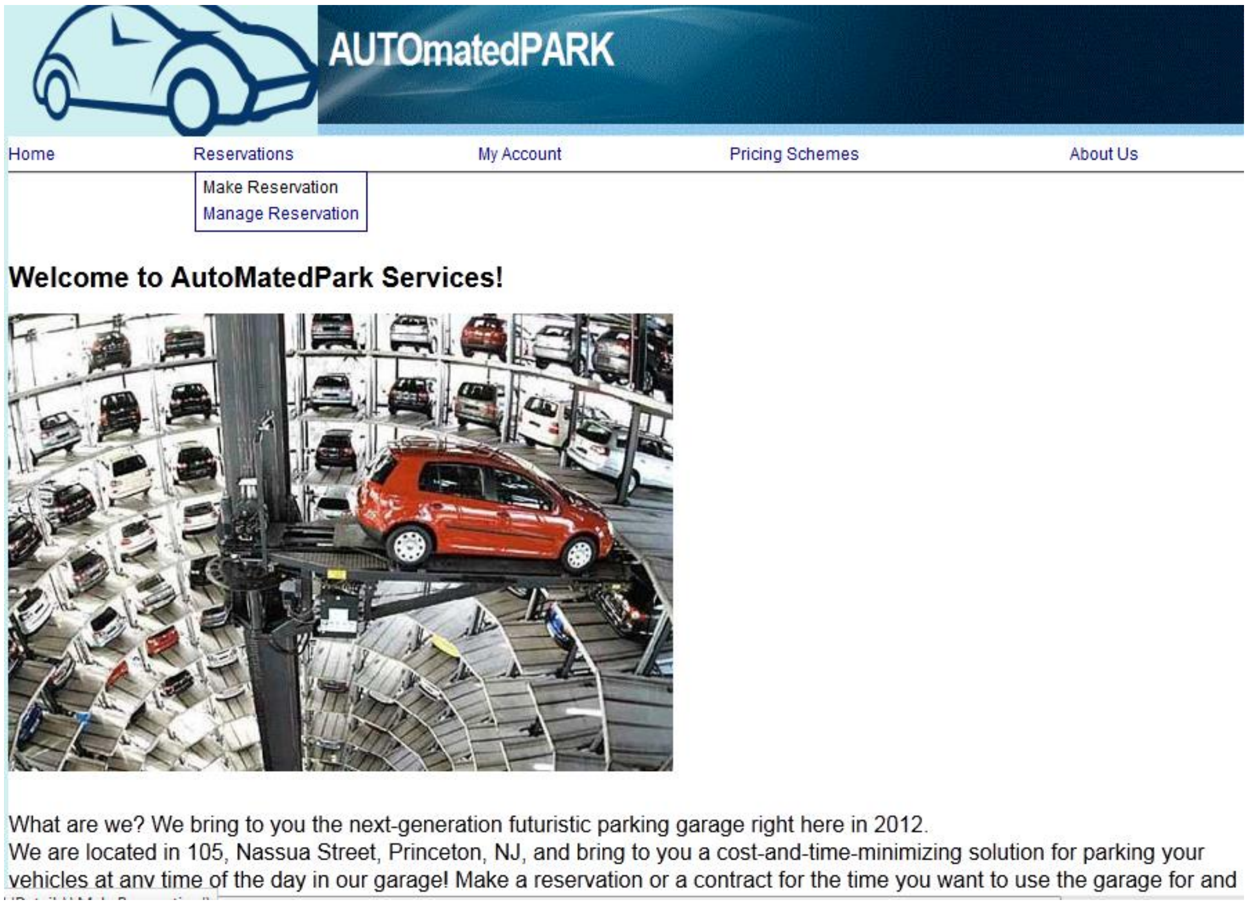
Homepage



Welcome to AutoMatedPark Services!

What are we? We bring to you the next-generation futuristic parking garage right here in 2012.
We are located in 105, Nassua Street, Princeton, NJ, and bring to you a cost-and-time-minimizing solution for parking your vehicles at any time of the day in our garage! Make a reservation or a contract for the time you want to use the garage for and

The initially proposed home screen was simpler and lacked a logo and picture. The tabs were also altered and categorized. Each of the “tabs” in the above screen is a hover-style menu, unlike the initial design. Ajax and jquery were used to create these menus.

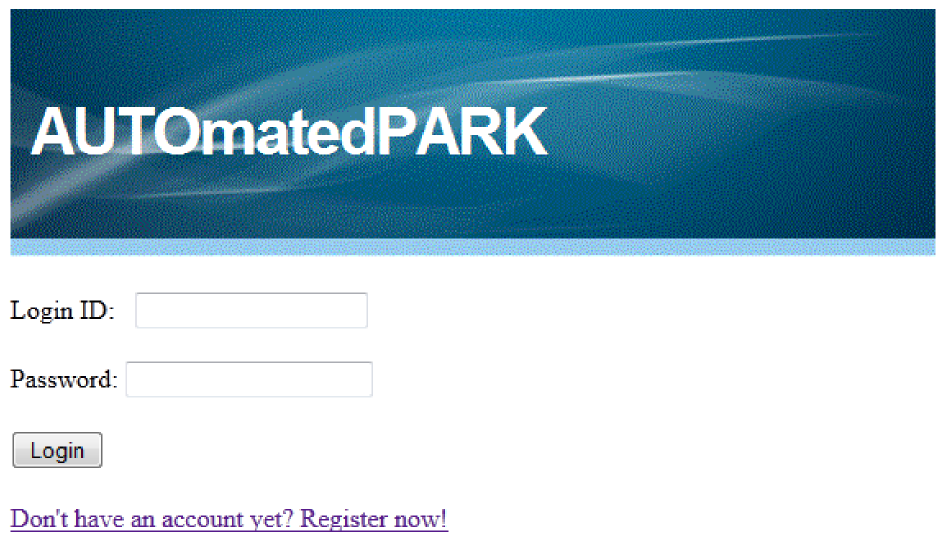


Welcome to AutoMatedPark Services!

What are we? We bring to you the next-generation futuristic parking garage right here in 2012.
We are located in 105, Nassua Street, Princeton, NJ, and bring to you a cost-and-time-minimizing solution for parking your vehicles at anv time of the day in our garage! Make a reservation or a contract for the time you want to use the garage for and

The new menus prevent the clutter of having multiple tabs on the screen. Managing and making reservations, managing user account, pricing information and contact information can be found without having to go through numerous tabs as in the previous design.

Login Page



Login ID:

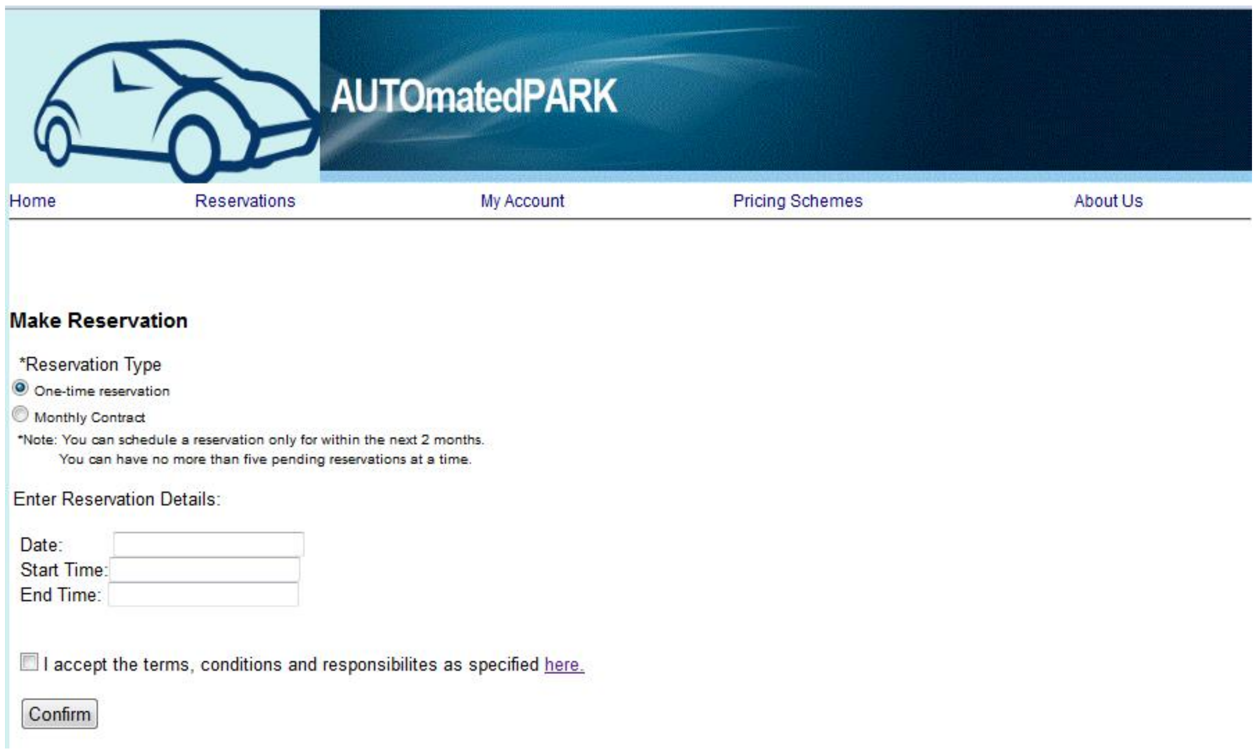
Password:

Login

Don't have an account yet? Register now!

The login screen has not changed significantly. The Register tab was eliminated and access to the registration page has been provided by a simple link under the login button.

Reservations



Make Reservation

*Reservation Type

(●) One-time reservation

( ) Monthly Contract

*Note: You can schedule a reservation only for within the next 2 months.
You can have no more than five pending reservations at a time.

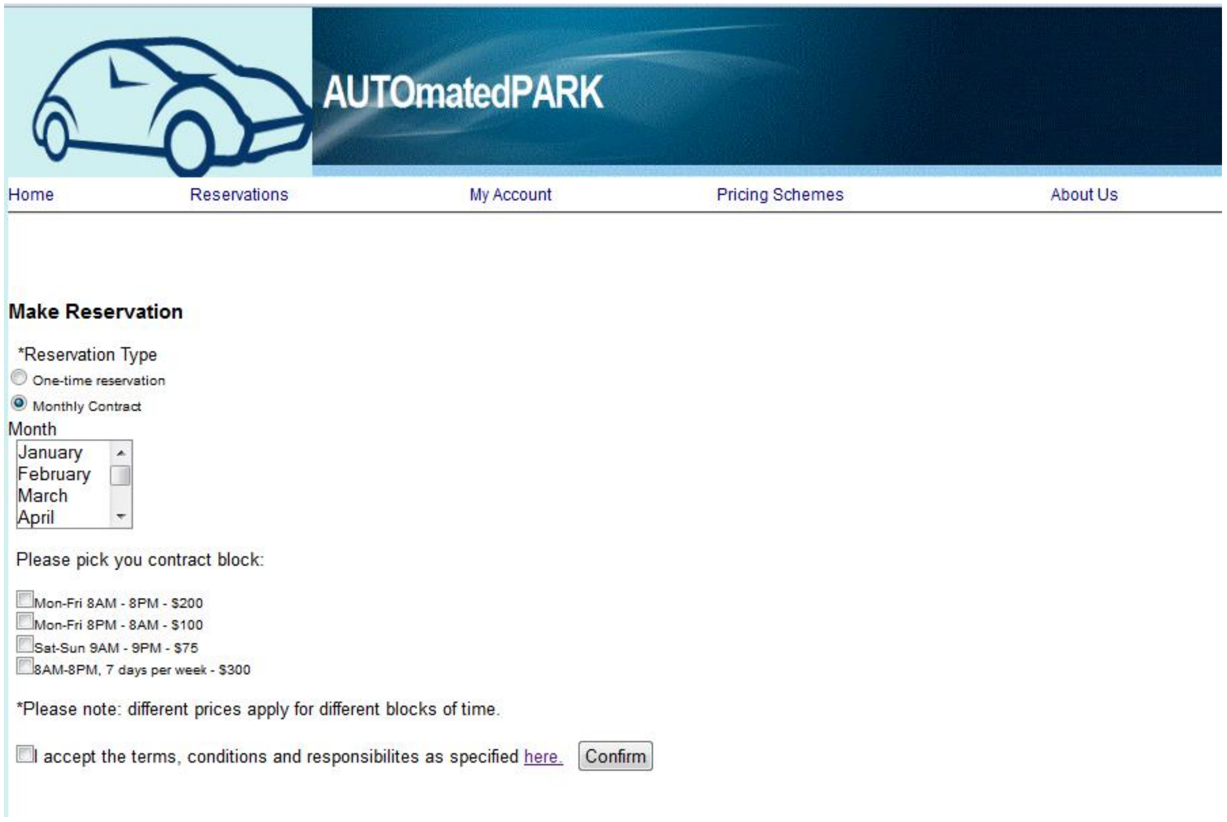
Enter Reservation Details:

Date:
Start Time:
End Time:

☐ I accept the terms, conditions and responsibilites as specified here.

Confirm

The make reservation screen of the initial design was again very simple. It had no dynamic pages which did not require a page refresh to change the layout.



Make Reservation

*Reservation Type

( ) One-time reservation

(●) Monthly Contract

Month

January
February
March
April

Please pick you contract block:

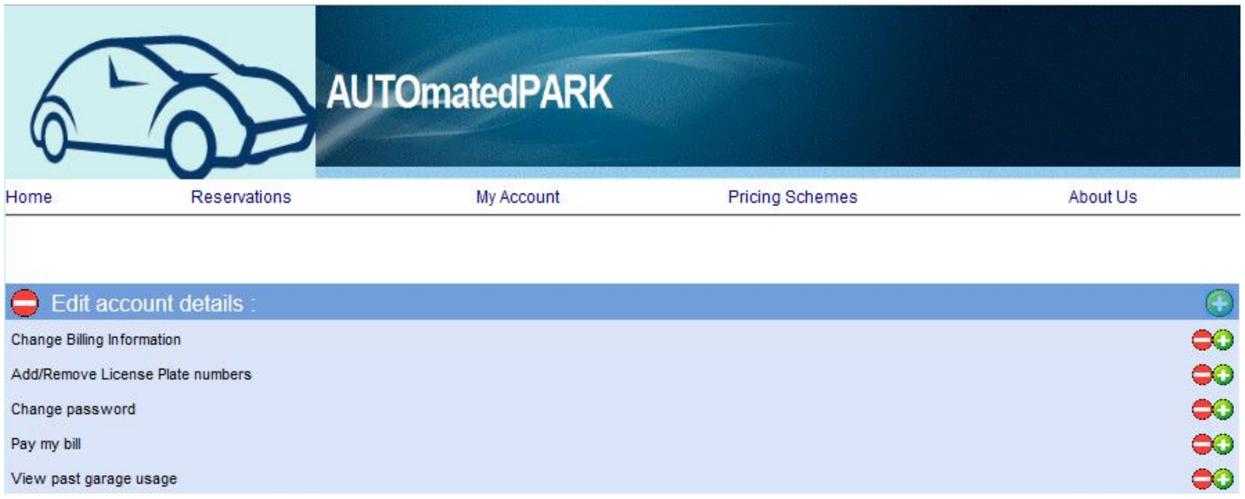
☐ Mon-Fri 8AM - 8PM - $200
☐ Mon-Fri 8PM - 8AM - $100
☐ Sat-Sun 9AM - 9PM - $75
☐ 8AM-8PM, 7 days per week - $300

*Please note: different prices apply for different blocks of time.

☐ I accept the terms, conditions and responsibilites as specified here. [Confirm]

As is visible, as each of the radio buttons is selected, the relevant information becomes visible to make a monthly or a one-time reservation.

My Account



Edit account details :

Change Billing Information

Add/Remove License Plate numbers

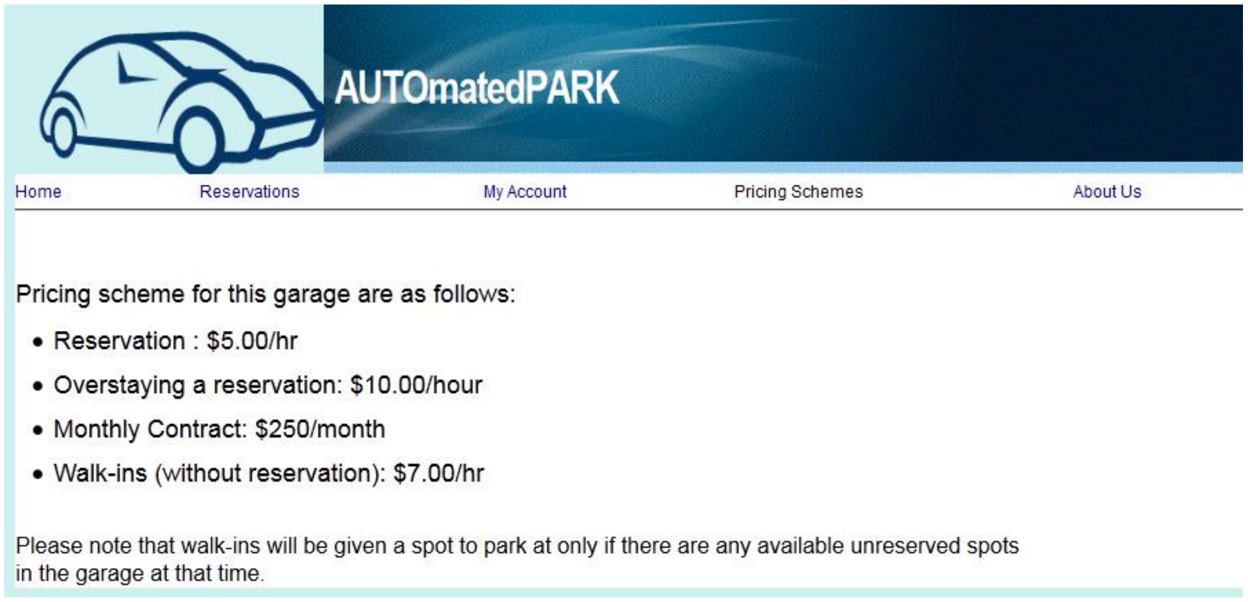
Change password

Pay my bill

View past garage usage

Compared to the initial user interface mock-up, the edit account page has been completely changed. In the plan that was drawn, this page would have many areas to optionally enter the information that required change. A password would have been mandatory to change any information. In the current design, the entire My Account page has been dedicated to manage the user account—including paying bills and viewing history. To the right of each option there is a red collapse button and a green expand button. When the expand button is clicked, the features necessary for a user to manage the relevant detail become visible. When the collapse button is clicked, the features are again hidden.

Pricing Schemes

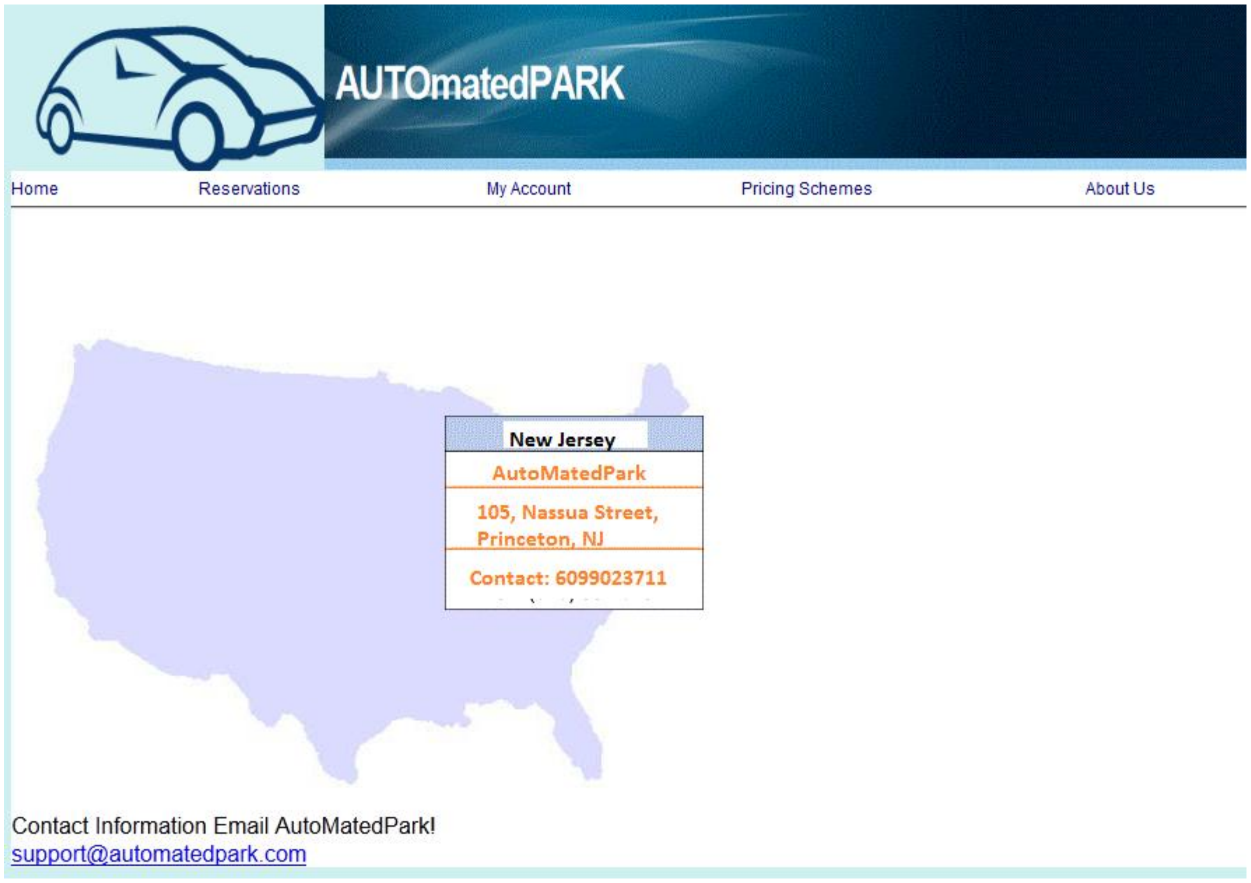


Pricing scheme for this garage are as follows:

- Reservation : $5.00/hr
- Overstaying a reservation: $10.00/hour
- Monthly Contract: $250/month
- Walk-ins (without reservation): $7.00/hr

Please note that walk-ins will be given a spot to park at only if there are any available unreserved spots in the garage at that time.

The pricing schemes tab is an entirely new concept that was not planned in the initial stages, but has been implemented.



New Jersey

AutoMatedPark

105, Nassua Street,
Princeton, NJ

Contact: 6099023711

Contact Information Email AutoMatedPark!
support@automatedpark.com

The above page is the About Us page. This page has also not been anticipated. It is new and creative.

13. Design of Tests

A. Unit testing:

Unit testing had to be done in three parts for our system because we had to check individually the functioning of the web pages, classes, and the database.

Webpage testing:

The functioning of the webpage was tested using the following steps.

- Once a user tries to access AUTOmatedPARK the first page to be displayed should be the login page.
- The control flow between the pages was tested to confirm that clicking on a particular button or making a particular selection takes the user to the required page.
- Clicking on the login button in the login page should take the user to the homepage.
- Since the goal was to implement hover menus cursors were moved over the tabs in the webpage to see if the sub list was displayed without having to click on it. Hover menus were implement using ajax.
- If the user clicks on make reservations he should be taken to the make reservations page.
- Our goal was to not refresh the entire page when switching between selections for which we used the update panel technique from ajax. If the user switches between the contract and reservation radio buttons under the make reservations page the entire page should not be refreshed. Rather just the section that contains the changing information should be refreshed.
- Once the user selects a contract term or reservation term and clicks on confirm, a popup should come up asking if the user accepts the terms and conditions. This is very critical because we need to make sure the users know about our policy.
- Once the user accepts to the terms the confirmation page should be displayed.
- If the user clicks on manage profile option under the My Account tab, the information that the user provided during registration should be displayed.
- If the user clicks on manage reservation under the My Account tab, information about the user's existing reservations should be displayed.
- In both the manage options the confirmation tab should be displayed after the user clicks finish.
- When the user clicks on the pricing information tab, the price info page should be displayed.
- If the user clicks on the edit account details option under the My Account tab, the edit account page should be displayed.
- The menus under the edit accounts page are designed to be collapsible to enhance user experience. This was done using jQuery and its functioning can be verified by clicking on

the green colored plus button to see if the options under a particular menu are being listed. To hide the detailed list of options the red colored minus button should be clicked and the list should collapse.
- When the user clicks on the pay button under the pay my bill option the payment confirmation page should be displayed. (In our current implementation clicking on pay will automatically display the payment confirmation page but in the actual system this would have to depend on the input received from the payment service provider such as paypal to see if the payment was valid.)
- If the user clicks on the About Us tab the location page should be displayed.
- If under the login page a user clicks on the new user option, the registration page needs to be displayed.

Thus the functioning of the webpage was tested by executing the above cases to see if we got the desired output. Linking the web pages is of the utmost importance because the user needs to be directed to the correct web pages and thus more effort was directed towards that direction.

Class testing:

We basically implemented a separate class to deal with each of the use cases. The testing for each of the classes was done by providing various inputs and checking if the procedure is able to distinguish between the correct and incorrect inputs.

Register:

If the register class is called with valid inputs for first name, last name, middle initial, email address, password, mailing address, credit card number, and phone number the system should call the addNewCustomer procedure in the database. It should then notify the page maker to display the confirmation page.

If the register function is called with invalid inputs such as the following then the procedure should prompt the user through the webpage for correct inputs.

Examples of invalid input that can be tested:

- Numbers or special characters in the first name, last name, or middle initial. (a blank middle initial or last name is acceptable)
- An email address without the @ symbol or with some non existing email provider.
- A password that is less than 8 characters and/or one without a number or a special character.
- A phone number that contains non numeric characters and/or not exactly 10 numbers.
- A mailing address with a non existing state and/or zip code.

- A credit card number that has non numeric characters.

The procedure has to keep prompting for valid inputs if it receives any of the above inputs until it receives a valid input.

Reserve:

The reserve class is called when the user picks a time period and clicks confirm. The following inputs could be given and checked to see if the class provides the desired output.

- For monthly contracts the user has to pick from one of the listed months and there is no possibility of invalid inputs in the case of contract. The addReservation procedure should be called in the database to add a new contract. The page maker should be called to display the confirmation page.
- In the case of time reservations there can be no invalid inputs for the date because the user has to pick from one of the listed months. If the input time period is more than 30 minutes the addReservation procedure should be called. The page maker should be called to display the confirmation page.
- If the input time period is less than 30 minutes then the user should be prompted for a longer time interval.
- If the input contains non-time characters the user should be prompted for valid input.

Enter Lift:

The enter lift class is called when the user actually arrives at the parking lot and the elevator senses the arrival of a user. This was tested by manually feeding the inputs for the various values that need to be sent to the system by the hardware components. The following inputs were given to the system and the output was analyzed. Since this is one of the major use cases we have a detailed list of test cases which are tabulated as follows:

Input	Output
1. Registered number plate is given as input for the license plate reader 1.	User should be identified as a reserved user
2. Contract user's number plate is given as the input for the license plate reader1.	User should be identified as a contract user
3. A plate number that does not exist in the reservation database is given as the input for the license plate reader1	The user should be asked if he is a registered or unregistered user
4. User inputs that he is registered	User should be asked for his login in details
5. User inputs valid login details	Check to see if he has a reservation or contract
6. User inputs invalid login details	Prompt for valid input
7. Unrecognized registered user has a	The user should be treated just like a regular

reservation or contract	reserved or contract customer
8. Unrecognized registered user doesn't have a reservation	he should be asked if he wishes to park as a walkin user
9. Unrecognized user inputs that he is an unregistered user	The user should be asked if he wishes to park as a walkin user
10. A walkin user agrees to the walkin terms	The user should be let into the ground floor.
11. A user is identified as contract	Transport user to the assigned floor.
12. A user is identified as reserved	Transport user to one of the upper levels if spot is available.
13. Reserved user arrives and upper levels are full	Check the database to see if there are vacant spots in the ground floor
14. Reserved user arrives when the garage is full	Request user to leave
15. A walkin user doesn't agree to the terms	Request user to leave
16. An unregistered walkin user agrees to walkin terms	A temporary account should be created for the duration of stay.
17. An unrecognized registered user is ready to park	The current plate number should be matched with the user's account for the duration of stay

Since this is one of the very important use cases a test application was created to test this part of the system by providing various inputs. This was presented during the demo.

Overstays:

The overstay class is called once every 30 minutes to detect users who are overstaying. The over stay class was tested using the following procedure.

- If a user is not checked out at the exit gate before the end of the reservation period this class should detect that the user is overstaying.
- If the overstay is detected for the first time then a text message has to be sent to the user warning him of the overstay policy.
- If a user is made to overstay for more than 30 minutes and overstay is detected for the second time, a text message should not be sent.

User Manage:

The user manage is called whenever the user wishes to make changes to his profile or his existing reservations. This class can be verified by feeding in the following inputs and seeing if we get the desired output.

Input	Output
1. A user who wishes to edit his profile provides valid input for all the fields as described in UC-1	The changes should be made in the database and reflected in the webpage.
2. A user who wishes to edit his profile provides invalid inputs examples of which are explained in UC-1	The user should be prompted for valid input details.
3. A reserved user wishes to cancel a reservation 30 minutes prior to the reservation start time	The reservation should be deleted from the database and a confirmation page should be displayed.
4. A reserved user wishes to cancel a reservation not 30 minutes prior to the reservation start time	The reservation is not deleted from the database and the user should be notified.
5. A reserved user wishes to extend a reservation prior to the end of the reservation	The database should be checked to see if there are available spots for the extended time period.
6. A reserved user wishes to extend a reservation after the end of the reservation	The request should not be processed and the user should be notified.
7. There are no spot available for the requested extension period	The user should be notified of the unavailability of spots.

Exit:

The exit class is called when a spot sensor senses a car leaving a spot. The exit class was tested by providing the following inputs and checking if we get the desired output.

Input	Output
1. Spot sensor input is given as vacant (state 0)	The type of leaving user should be determined.
2. A contract user is leaving	The spot should not be updated as vacant in the database.
3. A non contract user is leaving	The spot should be updated as vacant and the number of available spots in that floor should be increased by 1.
4. Spot senor input is not changed from state 1 to state 0.	The spot should remain as occupied in the database.

The administrative functioning of the system was given special attention because we wanted our product to be more appealing to the garage owners who are going to buy the actual product. An application was created to test the working of this part and was presented during the demo.

Color coded arrays were used to display the occupancy of the different floors of the garage at a particular time.

The admin should be able to get spot occupancy details such as the user who is currently occupying the garage, is the spot reserved for contract customer, etc by just clicking on a particular cell in the array.

If the admin inputs the number of floors in the garage and the number of spots each floor and select create the required number of floors with the required spots should be created and displayed.

If the admin wants to clear the screen he should be able to do it by clicking the “Clear” button. This should just clear the screen but should not destroy the object. To delete the created objects the “Destroy” button should be clicked.

The floor and spot details can be saved by selecting the “Save” button and it should give the option to save to a specified location. This will save the displayed details as a text file.

Database testing:

The working of the database can be checked if all the database procedures are functioning correctly. To check if a procedure we have to check if the tables that they are supposed to modify have been updated.

The following is a list of procedures and the tables they use or modify:

Procedure	Table
addNewCustomer	member, member_address, member_login
addReservationToDatabase	reserve_parking_online
deleteReservationFromDatabase	reserve_parking_online
getAmountDue	sales_Account

Sample testing code is as follows and the working of the procedure can be done by comparing the tables before and after execution of the test code.

<u>To test the addNewCustomer procedure</u>:

```
USE PARK

GO

execute addNewCustomer 'John','Benjamin',NULL,'1990-05-30','abcD',12435,'good ave east',NULL,'edison','nj','080','batman@gmail.com','batman@gmail.com','bestman23'
```

go

<u>To test the addReservationToDatabase procedure</u>:

```
USE PARK

GO

execute addReservationToDatabase 1, '2012-04-30T12:02:43.296',

'2012-03-29T12:12:43.296', 'abcD';

go
```

<u>To test the delteReservationFromDatabase procedure:</u>

```
USE PARK

GO

execute deleteReservationFromDatabase 1, '2012-04-30T12:02:43.296',

'2012-03-29T12:12:43.296';

go
```

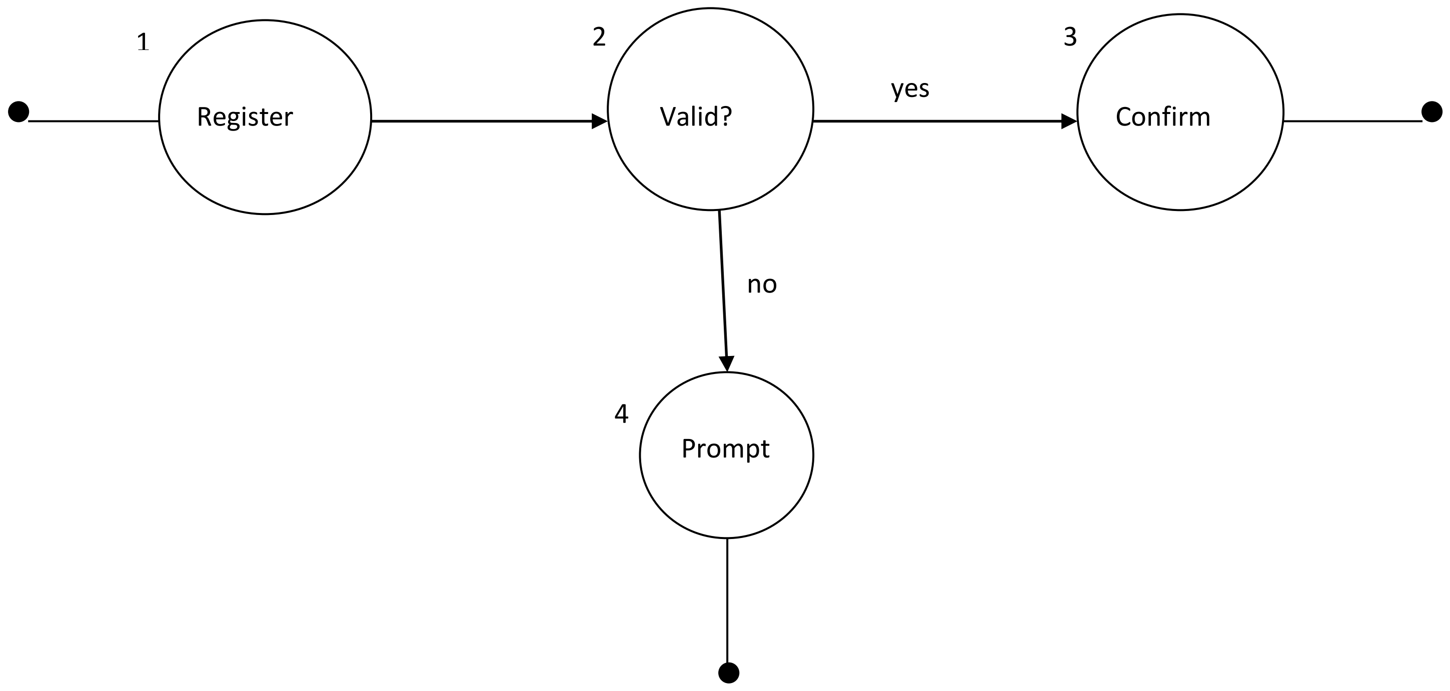
<u>To test the getAccountDue procedure:</u>

```
USE PARK

GO

execute getAmountDue 1

go
```

B. Test coverage:

Test coverage analysis was done using state diagrams. To ensure complete test coverage, all the states of the use cases have to be covered by at least one of the test cases. From the test cases stated above, we can ensure that all the states mentioned in the following state diagrams have been covered by at least one of the test cases.

Figure-34



States:

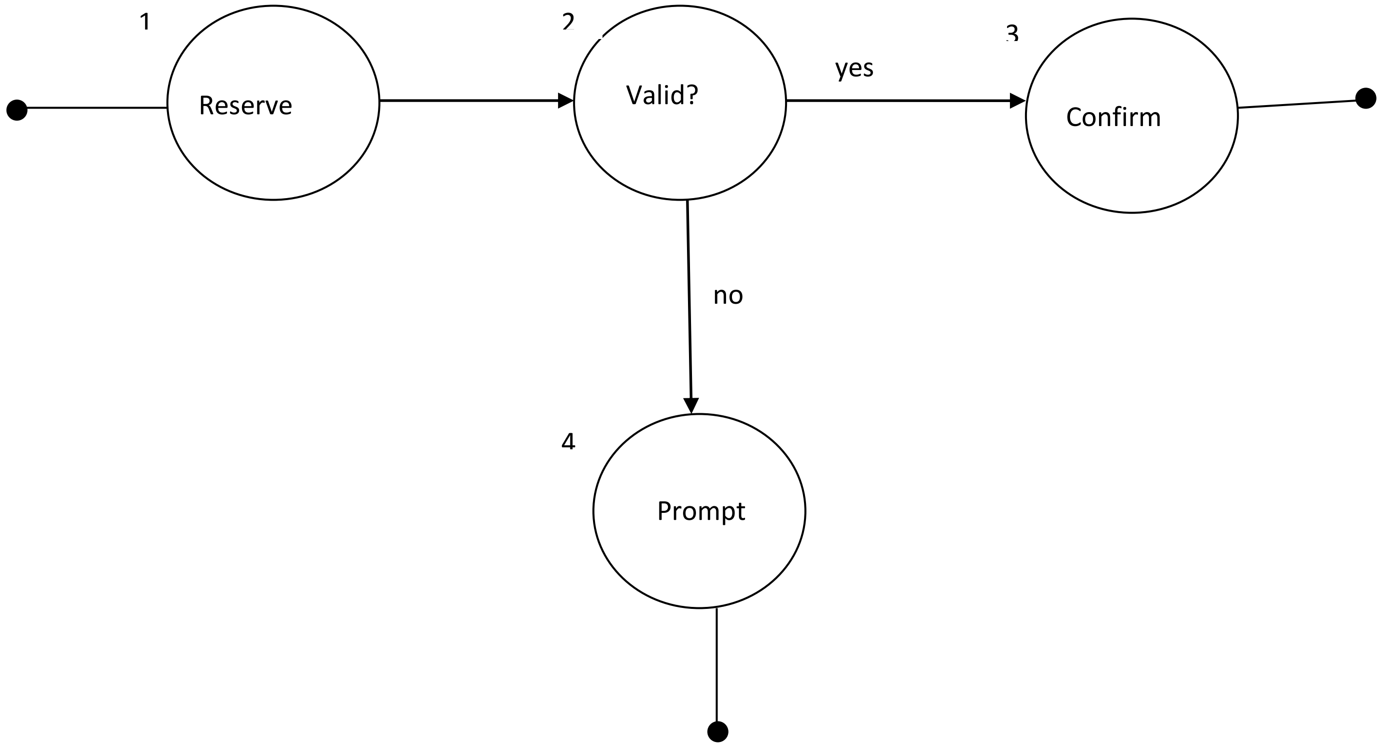
1-collect all inputs provided for registration

2-validate inputs

3-call page maker to display confirmation page

4-prompt the user for valid inputs

Figure-35



States:

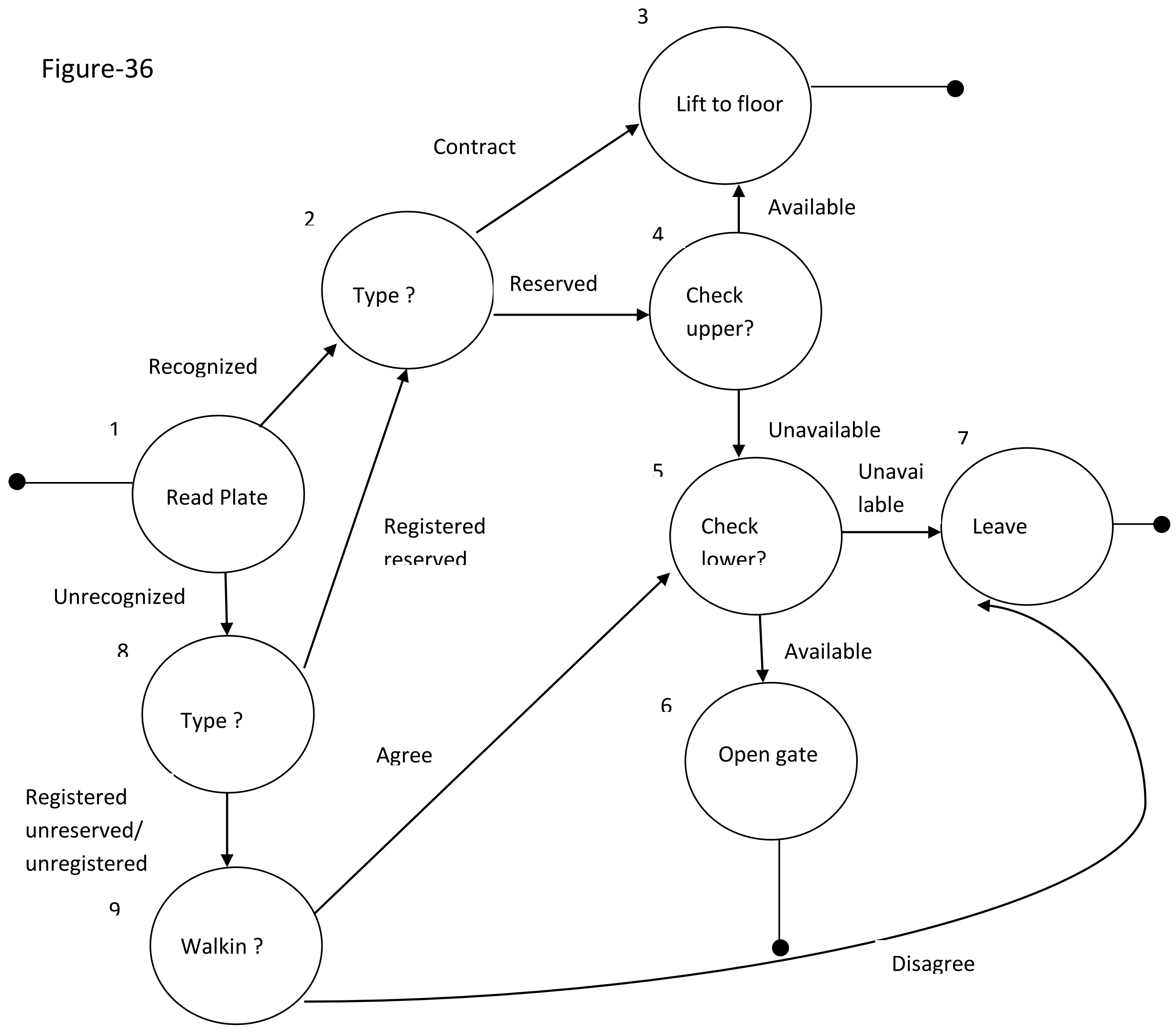
1-collect all inputs provided for reservation

2-validate inputs

3-call page maker to display confirmation page

4-prompt the user for valid inputs

Figure-36



States:

1-The license plate reader 1 is activated

2-determine if the user is reserved or contract

3-transport car to floor

4-check if spots are available in the upper levels

5-check if spots are available in the lower level

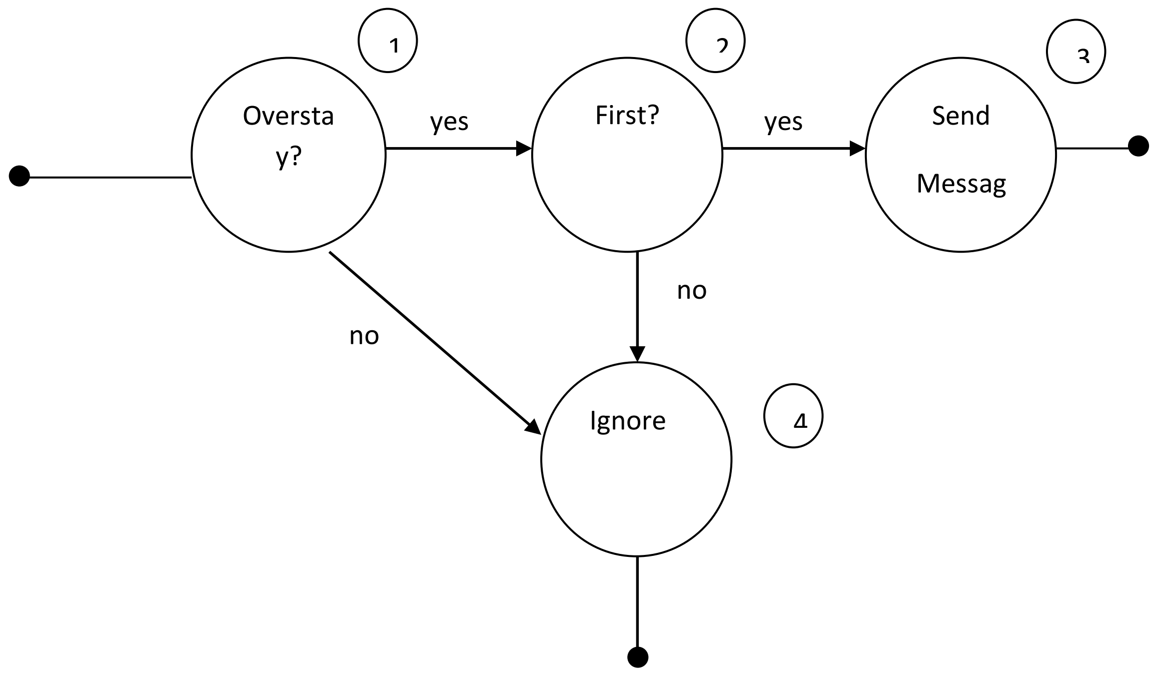
6-let in a user into the ground floor

7-request user to leave

8-determine type of user (ask for input)

9-agree to walkin terms?

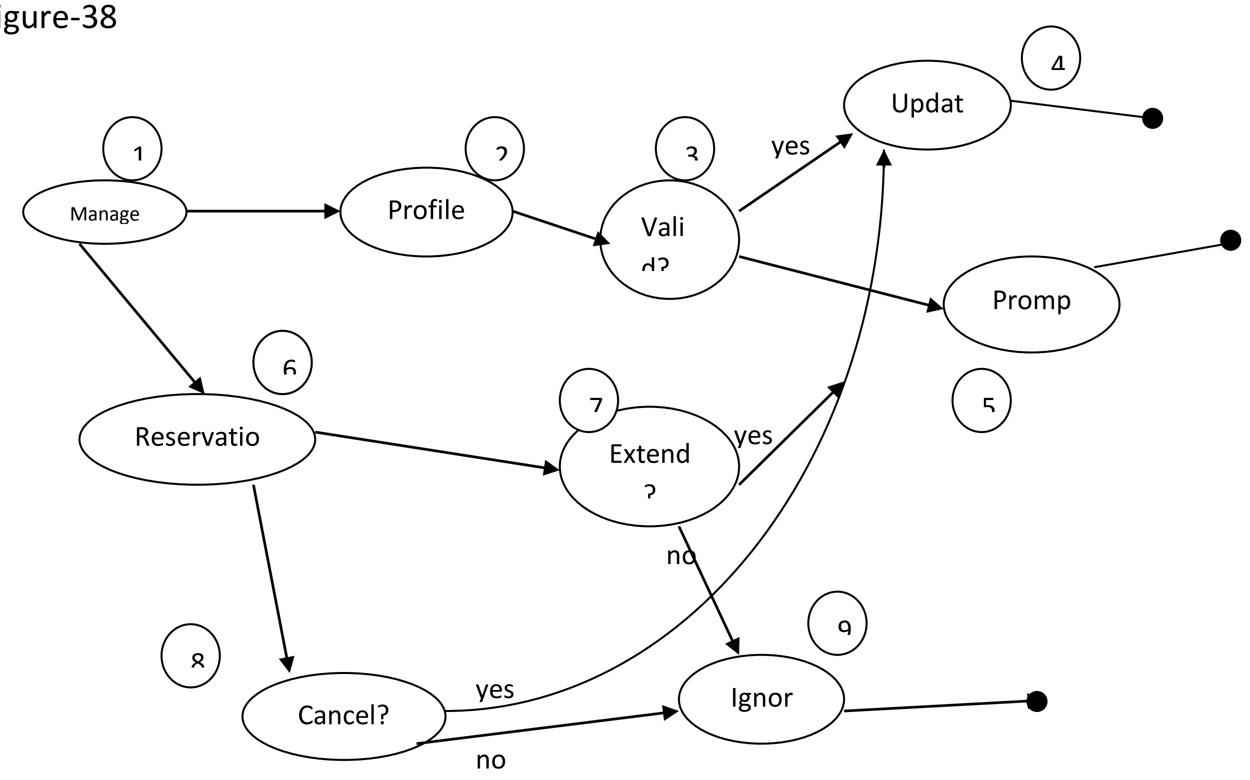
Figure-37



1. Call overstay checker
2. Is this the first time detection

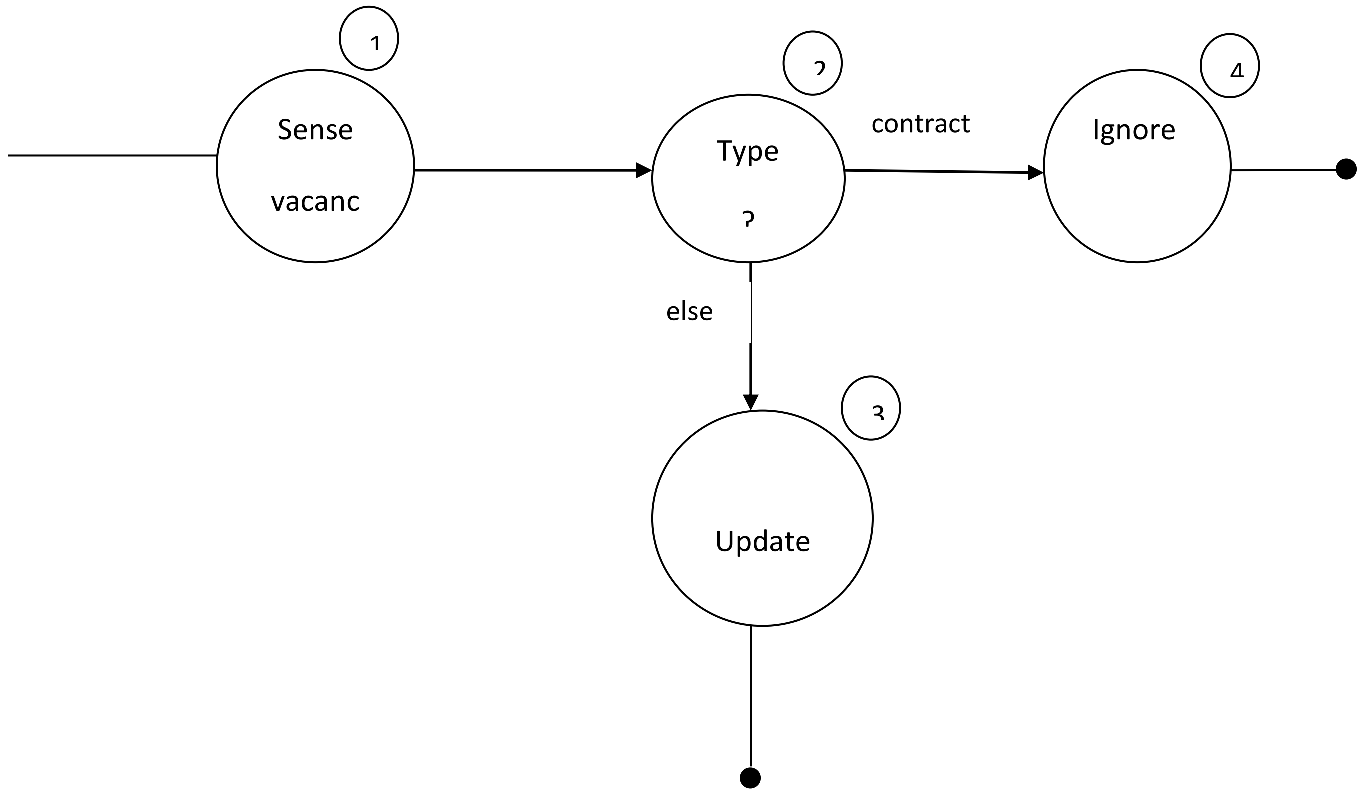
3. Send a waiting message
4. Take no action

Figure-38



1. Call user manage
2. Edit profile
3. Check for input validity
4. Update the database
5. Prompt for input
6. Manage reservation
7. Determine possibility to extend reservation
8. Determine possibility to cancel reservation
9. Take no action

Figure-39



1. Spot senor senses vacancy
2. Determine the type of user
3. Updated the database (mark spot available and increment number of available spots)
4. Take no action

C. Integration testing:

The main goal of integration testing was to make the three parts of the system namely the web pages, database and classes communicate appropriately with each other. “The purpose of integration testing is to verify functional, performance, and reliability requirements based on major design items.” It is used to identify potential bugs in system components as well as to confirm that they are interacting appropriately with one another. In our testing, we utilized what is referred to as Sandwich Testing. It combines Top Down testing that specializes in identifying missing links between modules or actors, with Bottom Up testing that is good at identifying small buds.

After the Sandwich Tests were completed, we implement what is referred to as Big Bang Testing. Big Bang Testing was used to confirm that the system runs properly as a whole in various use case permutations. This was done by simulating a realistic work environment in order to confirm that the system is properly configured to reliably, accurately, and efficiently handle all the scenarios that are required to meet specifications.

14. History of Work, Current Status, and Future Work

Major changes were made in the number of use cases that were presented in Report 1. Initially there were 14 use cases, but trivial use cases were eliminated and merged with other more significant ones. There are now a total of 8 use cases, one of which has not been implemented due to necessary third party involvement. The system sequence diagrams were also challenging to learn and create. They needed to be revised multiple times before deciding upon final versions. Another difficult aspect of this project was developing an encompassing domain model. Weeks and weeks of work had to be put in to decide which components needed to be included and which were unnecessary. The thing that was most discussed was the maintenance checker, which was added to check for mechanical failure in the devices that were needed to maintain the automated garage.

Report two was more challenging. The diagrams and design of tests were the most difficult parts. Making a complete class diagram required the group to think in a completely code oriented view, which was different from what was done in report one. Identifying subsystems was also difficult and making a UML package diagram out of the subsystems was challenging. The design of tests had to be done meticulously, to make sure that errors are not propagated into advanced stages of programming and make it infinitely more difficult to debug.

After most of the documentation and code was done it was easy to see the application of all that we were reading and learning of in class. This class was a great learning experience due to the experience that students received working in groups and dealing with different kinds of people. It was also nice to see the immediate and visible application of everything that was being taught in the classroom. Before coming to this class, we would have thought that finishing such a major project with a group full of such vibrant personalities was next to impossible. But, systematically, everything was coming together well and beyond expectation.

The most difficult thing about this class is working with a team and the sheer amount of effort it takes to stay in the group and function equally for the greater good of the team. Handling all of this while competing with the members in the team and members of the class is challenging.

Currently, the project is far from complete. There is a detailed view of information about the spots and its' occupants. The spots are color-coded based on spot status and they are clickable. There is an admin view where you can look at the history and status of occupancy. There is a hardware checker that can be connected to real devices and which will check for errors with the devices. There is also a customer look-up that enables the admin to search for customers by name. There is complete user manage code that enables users to change account information.

In the future it would be possible to arrange for bill processing via a third party like PayPal. Currently, the program does not support adjusting prices in administrative mode. Future work can include a billing rate setter. The customer search functionality is limited as of now; future work can include searching by license plate or phone number. There is also no mathematical model implemented to maximize occupancy and increase profits. There can also be customer and hardware logs and invoices to keep track of customers and devices that are in need of repair or replacement.

Key Accomplishments

- Developing use cases
- Experienced the agile method of development
- Working in a group
- Learned how documentation can help others and help with the coding process
- Analyzed the domain

References

1. [1] IEEE Computer Society, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems: IEEE Std 1472000. 2000.

2. Glossary definition:
http://msdn.microsoft.com/en-us/library/ee658098.aspx

3. Past class projects

4. Project descriptions

5. UI requirements pictures
https://transport.tamu.edu/account/NARequest/NAAccount.aspx
https://transport.tamu.edu/account/

6. System Sequence Diagrams:
http://www.ibm.com/developerworks/rational/library/3101.html
http://www.agilemodeling.com/artifacts/sequenceDiagram.htm

7. Russ Miles and Kim Hamilton: *Learning UML 2.0*, O'Reilly Media, Inc., Sebastopol, CA, 2006. (for UML Package Diagrams)

8. http://www.csharp-station.com/Tutorials/AdoDotNet/Lesson01.aspx (for Network Protocol)

9. http://www.agilemodeling.com/artifacts/packageDiagram.htm (for UML Package Diagrams)

10. "Integration Testing". http://en.wikipedia.org/wiki/Integration_testing Retrieved 3/11/2012

These links were used as reference to develop the interaction diagrams.

11. http://www.ibm.com/developerworks/rational/library/3101.html

12. www.visualcase.com/tutorials/interaction-diagrams.htm

13. http://www.agilemodeling.com/artifacts/sequenceDiagram.htm