GBC Parking Reservation Application

Software Requirements Specification

Version 1.0

11/11/19

Beatriz Morales
Afsana Bilkis Ritu
Jin Kwan Kim
Geng Zhang

Prepared for
COMP 3059—Capstone Project 1
Instructor: Anjana Shah
Fall 2019

Revision History

Date	Description	Author	Comments

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Beatriz Morales	Lead Developer	11-11-19
	Afsana Bilkis Ritu	Developer	11-11-19
	Jin Kwan Kim	Developer	11-11-19
	Geng Zhang	Developer	11-11-19

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the GBC Parking Reservation System. The document is split into four major sections: General Description, Specific Requirements, Analysis Models and Change Management Process. General Description includes the Product Perspective, Product Functions, User Characteristics, General Constraints and Assumptions/Dependencies. The Specific Requirements will include the various interfaces of the system, Functional Requirements, Use Cases, Classes/Objects, Non-Functional Requirements, Inverse Requirements, Design Constraints and Logical Database Requirements. The Analysis Models will include the Sequence Diagrams, UML Diagram, Activity Diagrams, Normalized Data Model Diagram and DFD Diagrams. Lastly, the Change Management Process will explain the process of making changes to this document should the need arise. This document is intended for both the stakeholders and the developers of the system.

1.2 Scope

This software system will be a web reservation system for George Brown College students and professors. This system will be designed to reduce the time spent looking for a parking space by allowing the user to reserve a space online and provide the option to pay online as well. Additional functionalities will include notifications, reservation cancellation, comments submission, payment history, reservation extension, view comments, pay late fees, create user account and view the list of users. By reducing the time spent looking for a parking space the system will meet the needs of the users, which will in turn reduce their frustration and allow them more time to focus on important things, such as their studies. The system will contain a database which will contain a list of users, reservations, payment information and admin information.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
User	GBC student or professor who will be using the system
Admin	The system administrator who will oversee the system
GBC	Short for ‘George Brown College’
Database	Collection of all the information pertaining to this system
Software Requirements Specification	A document that completely describes all the functions of a proposed system and the constraints under which it must operate. For example, this document.
Stakeholder	Any person with an interest in the project who is not a developer.

1.4 Overview

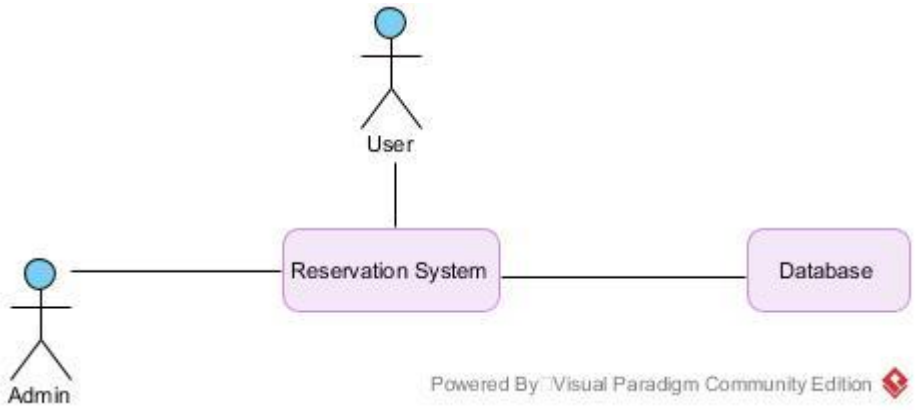
The next chapter, the General Description section of this document gives an overview of the functionality of the application. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter. The third chapter, the Specific Requirements, is written primarily for the developers and describes in technical terms the details of the functionality of the product. The fourth section, the Analysis Models, contain the diagrams which will outline how the system will work in terms of sequences, processes and database. The fifth and final section of this document, the Change Management Process, will outline the procedure if changes need to be made to the project scope and requirements.

.

2. General Description

2.1 Project Perspective

System Environment



The reservation system has two active actors and one cooperating system. The users and admin access the parking reservation system through the internet. The system pulls data from the database in order to provide the information needed to successfully run a function. This is a brand-new system designed for the purpose of the project and aims to solve the issue that students and professors of George Brown College face when trying to find a parking spot.

2.2 Product Functions

The use cases will be outlined in section 3.3. User has nine use cases, while admin has two.

2.3 User Characteristics

The user is expected to be internet literate and be able to access the system, as well as being able to use drop down menus and buttons to navigate the site. Additionally, users should be comfortable using email as notifications will send an email if they opt in. The admin is expected to be comfortable with the system features and be able to navigate the site easily. They should be able to use drop down menus' buttons and tables effectively as these will be needed to be able to pull user information.

2.4 General Constraints

Software Limitations:

- Due to the nature of the project, being a school project, and our status as students, there is a limit to the type of software we can use to create the system.

Time Constraint:

- Fixed time limit to complete the project (7 months, from September 2019-March 2020)

Technical Constraint:

- System should be able to accommodate many users

Business Constraint:

- Small team, only four developers working on this project

2.5 Assumptions and Dependencies

- The final version of the application will include the functionalities outlined in the project vision document
- The project will serve its intended purpose, which is to assist GBC students and professors in reserving parking spaces
- Required programming libraries and IDEs will be available to use
- End users will be able to test the application

3. Specific Requirements.

3.1 External Interface Requirements

There is no link to an external system for the application since it will be getting information from the database to the system and then sending it to the user or admin, depending who is requesting information.

3.1.1 User Interfaces

All pages of the application will follow a standard layout containing back or cancel buttons, depending on which page it is. For example, a multi-step function such as reserving a parking spot would involve the implementation of back buttons in the event that the user wants to backtrack to a previous step, while a single step function such as cancelling a reservation would have a cancel button if the user wishes to terminate the process. All the pages will also have a logout button on the top right corner where the user can logoff the system. The user and admin dashboards will follow a similar layout where the various functions will be shown at the top of the page via clickable buttons (submit comment, reserve spot, view comments etc).

3.1.2 Software Interfaces

The system will make use of a database to hold all the necessary information for the application to function. The database will contain four tables: User, Admin, Reservation and Payment Information. The user table will contain the user id, first name, last name, email, license plate number, password, reservation id, payment id and user comment. The admin table will contain the admin id and password. The reservation table will include the reservation id, date, user id, lot number and spot. Lastly, the payment information table will contain the payment id, user id, first name, last name, card number and payment date.

3.1.3 Communications Interfaces

Email will be used to send the users receipts and notifications. The email used will be the one entered in the database when the user signs up.

3.2 Use Cases

3.2.1 Use Case #1: Parking Spot Selection

Primary Actor: User
Level: Kite
Stakeholders: Users, Admin
Precondition: User creates account and logins to system
Success guarantees: Reservation confirmed
Trigger: User logins to the application
Main Success Scenario:

1. User clicks on the “reserve a parking spot” button in the user dashboard
2. User selects desired parking lot
3. User selects spot and time
4. If the spot & time slot is available, the user can proceed with entering their details
5. User enters license plate number and name
6. Reservation is logged in the system
7. The system will send the user an email with the reservation details
8. Reservation is updated in the system, shows the spot is taken

Extensions:

1. Reservation system is unavailable
   - User exits out of the system
2. Desired parking lot is full
   - User selects a different lot

OR

- User exits out of the system

3. Desired parking spot & time is taken
   - User selects a different spot

   OR
   - User exits out of the system
4. User changes their mind about the lot
   - User can click on the “back” button to return to the lot selection
5. User changes their mind about the parking space and time
   - User can click on the “back” button to return to parking space selection
6. User enters invalid license plate and name information (e.g, all numeric or all characters)
   - Error message will show saying that the input is invalid
7. User changes their mind about the reservation; they do not wish to reserve a spot
   - User can click on the “cancel” button to return to the user dashboard page

3.2.2 Use Case #2: Parking Reservation Cancellation

Primary Actor: User
Level: Kite
Stakeholders: Users, Admin
Precondition: User reserves a parking spot and is sent email confirmation
Success guarantees: Reservation is cancelled
Trigger: User accesses the “user dashboard” page
Main Success Scenario:

1. User clicks on the “cancel reservation” button in the user dashboard
2. User confirms cancellation of the spot
3. The system will remove the reservation

Extensions:

8. Reservation system is unavailable
   - User exits out of the system
9. User does not have a reservation to cancel
   - System will show a message to user telling them that they do not have a reservation
10. User changes their mind and wishes to keep their reservation
    - User can click on the “cancel” button to go back to the user dashboard page

3.2.3 Use Case #3: Parking Reservation Payment

Primary Actor: User
Level: Kite
Stakeholders: Users, Admin
Precondition: User reserves a parking spot and is sent email confirmation
Success guarantees: Payment is successful
Trigger: User reserves a parking spot
Main Success Scenario:

1. User clicks on the “pay for parking reservation” button after successfully reserving a spot

2. User enters in their payment information
3. The system will process the payment and display a success message for the user letting them know that the payment was successfully processed
4. System will email user a receipt displaying the amount payed for the spot

Extensions:

5. Reservation system is unavailable
   - User exits out of the system
6. User enters invalid payment information (non-numeric credit card number, numeric string for name, etc.)
   - System will show an error message telling the user that the information is invalid
7. User changes their mind and does not wish to pay for their reservation online
   - User can click on the “cancel” button to go back to the user dashboard page

3.2.4 Use Case #4: User information

Primary Actor: Admin
Level: Kite
Stakeholders: Admin
Precondition: Users have created an account on the system
Success guarantees: User information is displayed
Trigger: Admin accesses the user information page
Main Success Scenario:

1. Admin clicks on the “user information” button in the admin dashboard page
2. Admin selects the user they wish to see the information of from a drop-down menu
3. System will retrieve the selected user’s information from the database and display it for the admin to see
4. Admin can view the username, license plate number, email, reservation (if any), payment information

Extensions:

5. Reservation system is unavailable
   - Admin exits out of the system
6. Admin wants to view a different user
   - Admin can click the drop-down menu and select another user. The system will retrieve and display the user’s information
7. Admin wants to quit looking at users
   - Admin can click the “cancel” button to return to the admin dashboard screen

3.2.5 Use Case #5: Notification of reservation expiration

Primary Actor: User
Level: Kite
Stakeholders: User
Precondition: Users have reserved a spot
Success guarantees: 15 minutes before expiration reminder is emailed to user
Trigger: User reserves a parking spot
Main Success Scenario:

1. After user reserves a spot system will ask user if they want a reminder emailed to them 15 minutes before their reservation expires
2. User clicks “yes” button

3. User will receive an email with the reminder that the reservation is going to expire 15 minutes before the expiration time

Extensions:

4. Reservation system is unavailable
   - User exits out of the system
5. User does not want notification emailed to them
   - User clicks “no” on the prompt

3.2.6 Use Case #6: Payment History

Primary Actor: User
Level: Kite
Stakeholders: Users
Precondition: Users have paid for a reservation at least once
Success guarantees: Page will display the payment history of the user
Trigger: User accesses the payment history page
Main Success Scenario:

1. User clicks on the “payment history” button on the user dashboard page
2. Page will display with a table outlining the date and the amount paid

Extensions:

3. Reservation system is unavailable
   - Admin exits out of the system
4. User has never paid for a reservation
   - Page will display an empty table

3.2.7 Use Case #7: Extension of reservation

Primary Actor: User
Level: Kite
Stakeholders: User
Precondition: Users have an active reservation
Success guarantees: Reservation time is extended up to a maximum of 2 hours
Trigger: User accesses the extend reservation page
Main Success Scenario:

1. User clicks on the “extend reservation” button on the user dashboard
2. Page will display the current reservation and give an option to extend to the time up to 2 hours
3. User will select the amount of time they wish to extend the reservation up to a maximum of 2 hours
4. System will update the reservation with the new duration as specified by the user

Extensions:

5. Reservation system is unavailable
   - User exits out of the system
6. There is another reservation after the original time of the user’s reservation
   - User will receive a message saying that they cannot extend the reservation time because there is already another reservation after them
   - User can either make a new reservation on a different spot or find another parking space manually
7. User changes their mind and does not want to extend the reservation
   - User can click the “cancel” button to go back to the user dashboard page

3.2.8 Use Case #8: Payment of late fees

Primary Actor: User
Level: Kite
Stakeholders: Users
Precondition: Users have an active reservation and park over their time limit
Success guarantees: User will receive a list of fees to pay and pay them
Trigger: User reservation time goes over the reserved limit
Main Success Scenario:

1. User can click on “pay late fees” button to access the late fees page
2. Late fees page will display the fees owed
3. Users can click on the “pay now” button to access the payment page
4. Users can enter in their payment information
5. System will process the payment and display a success message
6. User will be emailed a receipt of the payment

Extensions:

7. Reservation system is unavailable
   - User exits out of the system
8. User does not have any late fees to pay
   - Page will display as empty
9. User changes their mind and does not wish to pay their late fees now
   - User can click the “cancel” button to go back to the user dashboard page

3.2.9 Use Case #9: Submit Comments

Primary Actor: User
Level: Kite
Stakeholders: Users
Precondition: Users have created an account on the system
Success guarantees: User will fill out a comment and submit it
Trigger: User access the submit comment page
Main Success Scenario:

1. User clicks on the “submit comment” button on the user dashboard page
2. Page will display a form
3. Users can fill out their name, GBC number and comment they wish to submit
4. User will click the submit button to submit the comment
5. Comment will be stored in the database where admin can view it

Extensions:

6. Reservation system is unavailable
   - User exits out of the system
7. User does not fill in their name
   - Error message will display saying that name is required
8. User does not fill in their GBC number
   - Error message will display saying that GBC number is required
9. User does not fill out a comment(s)
   - Error message will display saying that a comment is required
10. User changes their mind and does not wish to submit a comment
    - User can click the “cancel” button to go back to the user dashboard page

3.2.10 Use Case #10: View comments

Primary Actor: Admin
Level: Kite
Stakeholders: Admin, Users
Precondition: User has submitted a comment
Success guarantees: Admin can view a list of users and their comments
Trigger: Admin accesses the view comments page
Main Success Scenario:

1. Admin clicks on the “view comments” button on the admin dashboard page
2. Page will display table with only the users that have submitted comments along with their actual comment

Extensions:

3. Reservation system is unavailable
   - Admin exits out of the system
4. There are no users that have submitted comments
   - Page will display empty table
5. Admin does not wish to view user comments anymore
   - Admin can click the “back” button to return to the admin dashboard page

3.2.11 Use Case #11: Create Account

Primary Actor: User
Level: Kite
Stakeholders: User
Precondition: None
Success guarantees: User can create an account and login
Trigger: User clicks on the ‘Create Account’ button
Main Success Scenario:

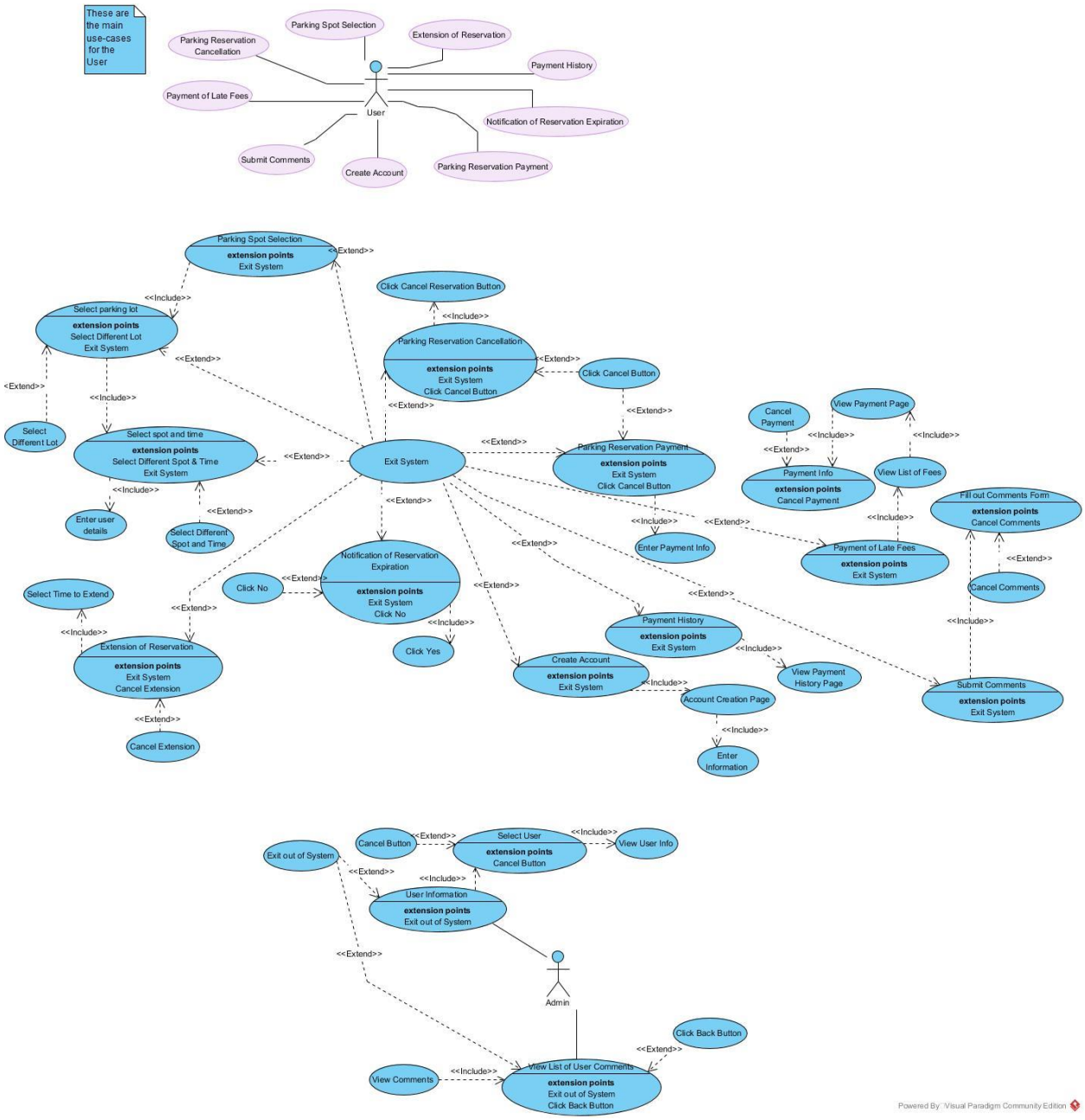
1. User clicks on the “create account” button on the login page
2. System will display the account creation page
3. User enters their name, GBC number and email and clicks the ‘submit’ button
4. System will create the account for the user and the user can then login to the system

Extensions:

5. Reservation system is unavailable
   - User exits out of the system
6. User enters invalid GBC number (string instead of integers)
   - System will show an error message saying to enter numbers
7. User enters invalid name (numbers instead of a string)
   - System will display an error message saying to enter a string
8. User enters an invalid email address (gsfsfs@hotmail..com , extra dot before com)
   - System will show error message to please enter a valid email)

3.2.12 Use Case Diagram

Below is the use case diagram. The way the diagram is structured is as follows: the nine use cases for the user are displayed and the main success scenario steps are shown as ‘”include” relationships, while the extensions are shown as “extend” relationships. The same applies for the admin’s use cases. A smaller, additional diagram is included on top of the user’s use case diagram to outline the main use cases for the user for ease of clarity.



3.3 Classes / Objects

3.3.1 <User>

3.3.1.1 Attributes:

- userId: integer that represents that user's GBC number
- firstName: string that represents the user's first name
- lastName: string that represents the user's last name
- email: string that represents the user's email address
- licensePlateNumber: string that contains the license plate number entered by the user when making a reservation
- userPswd: string that represents the user's account password
- userComment: a string that contains the user's comment

3.3.1.2 Functions

- User: the constructor of the user class; contains all the attributes mentioned above
- toString: combines the parameters into a string.

3.3.2 <Admin>

3.3.2.1 Attributes:

- adminId: integer that represents the admin's id number, used to sign into the system
- adminPassword: string that contains the admin's password to sign into the system

3.3.2.2 Functions

- viewComment: function that will pull the user comments from the database and display it for the admin to view
- viewUserInformation: function that will take the user's information from the database and display it for the admin to view

3.3.3 <User Manager>

3.3.3.1 Attributes:

- currentUserNo: an integer that act as a pointer to keep track of the current selected user
- numUsers: an integer that counts the amount of users
- userList: an array of users that contains all of the attributes of the user table

3.3.3.2 Functions:

- User_Manager: the constructor of the User Manager class
- addUser: adds the user id, first name, last name, email and password to the user array
- findUser: search function using the user id to locate a specific user
- userExist: checks if the user exists. Returns true or false
- getUser: uses the user id to return the user's information stored in an array

- getUserList: returns a list of users
- submitComment: stores the user comment as a string

3.3.4 <Reservation>

3.3.4.1 Attributes:
- reservationId: integer that identifies a reservation as unique
- date: string that contains the date of when a reservation was created
- lotNumber: integer that represents which lot a reservation pertains to
- spot: string that contains the spot name

3.3.4.2 Functions:
- Reservation: the constructor of the reservation class
- toString: combines the reservation attributes into a string
- addClient: adds the user information to the reservation, returns true or false
- removeUser: removes a user from a reservation using the user id, returns true or false

3.3.5 <Reservation Manager>

3.3.5.1 Attributes:
- numReservations: an integer that represents the current number of reservations in the system
- reservationList: an array containing the list of reservations

3.3.5.2 Functions:
- Reservation Manager: the constructor of the Reservation Manager class
- addReservation: adds a reservation to the reservation array; returns true or false
- findReservation: attempts to locate a reservation using the reservation id; returns true if found, false otherwise
- reservationExist: checks if the reservation exists; returns true or false
- getReservation: Retrieves the reservation based on the id
- updateReservation: extends the reservation time; requires the reservation id
- cancelReservation: deletes the reservation using the reservation id
- User_Manager: allows for the use of the user attributes in order for the other functions to work

3.3.6 <Payment Information>

3.3.6.1 Attributes:
- paymentId: integer that represents the unique id number attached to a payment
- cardNumber: represents the user's credit card number
- pymentDate: string representation of the date a payment was made

3.3.6.2 Functions:

- User_Manager: imports the user information needed for a payment (id, email, first name and last name)
- generateReceipt: creates a receipt and sends to the user's email

3.4 Non-Functional Requirements

3.4.1 Reliability

The system should aim to have a close to 100% uptime so that more GBC students and professors feel inclined to use the application. The more reliable the system is, the more users it will attract, and the more users it has the greater chance it has to attain the systems main purpose: which is to solve parking problems and reduce frustration.

3.4.2 Maintainability

The system should be able to be maintained over a space of several years or more. If an error occurs that causes the system to cease functions, the developers should be able to fix the issue and have the system up and running again in a reasonable amount of time.

3.5 Logical Database Requirements

A database will be used to store all the information pertaining to the system. The tables below outline the data entities and their attributes.

User Data Entity

Data Item	Type	Description	Comment
User Id	Integer	Id number of the system user	Primary key
First Name	Text	The user's first name	
Last Name	Text	The user's last name	
Email	Text	The user's email address	
License Plate Number	Text	The user's license plate number	
User Password	Text	The user's password	
Reservation Id	Integer	Identification Number for reservation	Foreign key
Payment Id	Integer	Identification number for a payment	Foreign key
User Comment	Text	The user's comment	

Reservation Data Entity

Data Item	Type	Description	Comment
Reservation Id	Integer	The reservation's identification number	Primary key
Date	Datetime	The date of the reservation	
User Id	Integer	Id number of the system user	Foreign key
Lot Number	Integer	The parking lot's number	
Spot	Text	The name of the parking spot	

Payment Information Data Entity

Data Item	Type	Description	Comment
Payment Id	Integer	The identification number for a payment	Primary key
User Id	Integer	Id number of the system user	Foreign key
First Name	Text	The user's first name	
Last Name	Text	The user's last name	
Card Number	Integer	The user's credit card number	
Payment Date	Datetime	The date the payment was made	

Admin Data Entity

Data Item	Type	Description	Comment
Admin Id	Integer	The identification number for the admin	Primary key
Admin Password	Text	The admin's password	

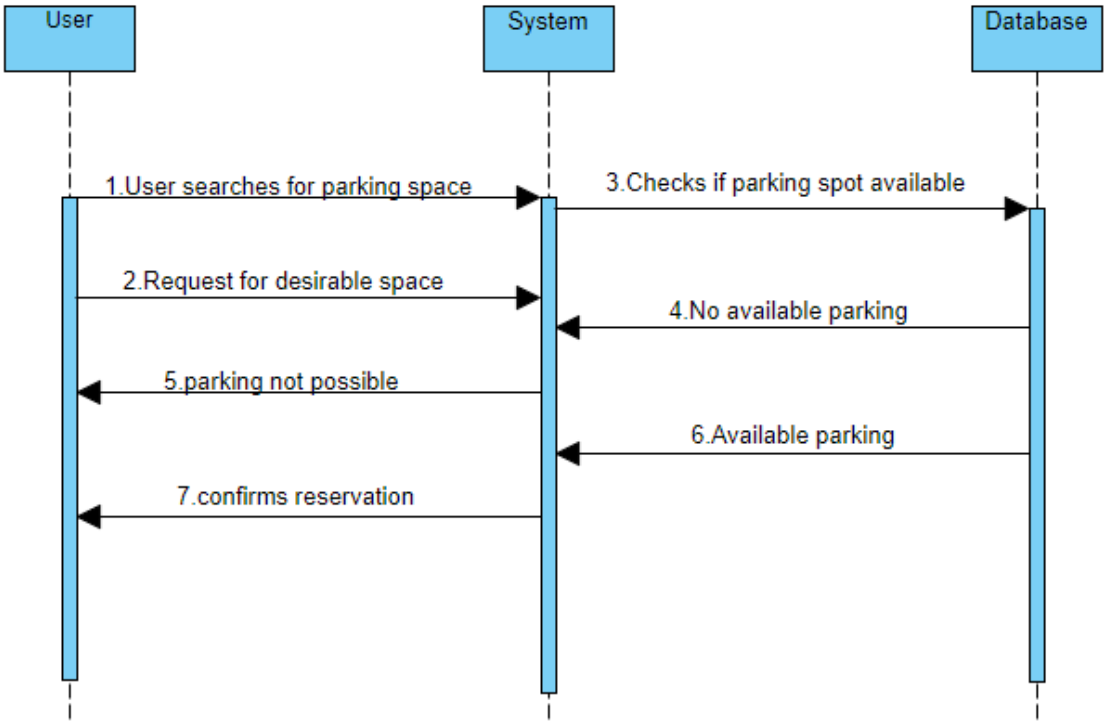
4. Analysis Models

4.1 Sequence Diagrams

Below are the sequence diagrams for the application. There are eleven total diagrams, each pertaining to a use case. Additional information will be provided above each subsequent diagram.

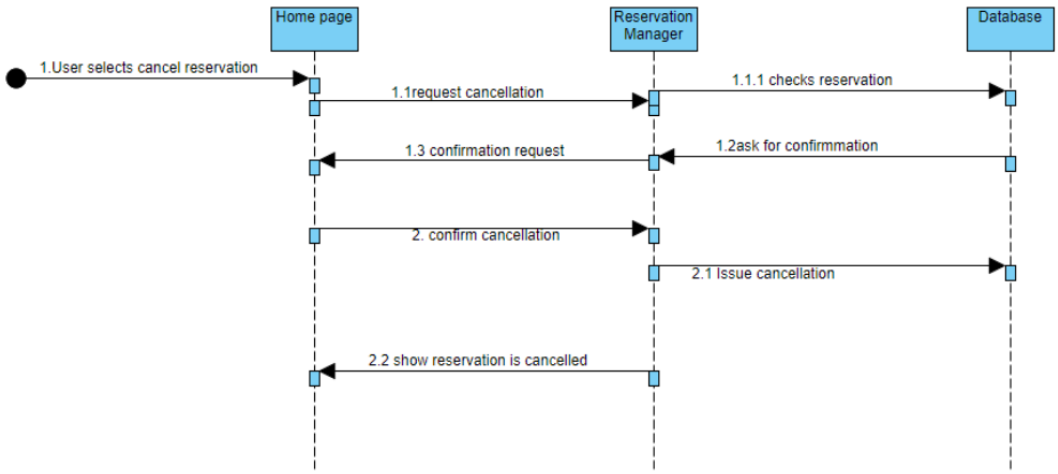
4.1.1 Parking Spot Selection

This diagram illustrates the sequence of processes pertaining to reserving a parking spot. First the user searches for the parking space they would like and submits their request to reserve that spot. The system receives the request and then checks the database to see if the spot is available. If the spot is not available, then the system will issue a message to the user saying that the spot is unavailable. If the spot is available, the user can reserve the spot.



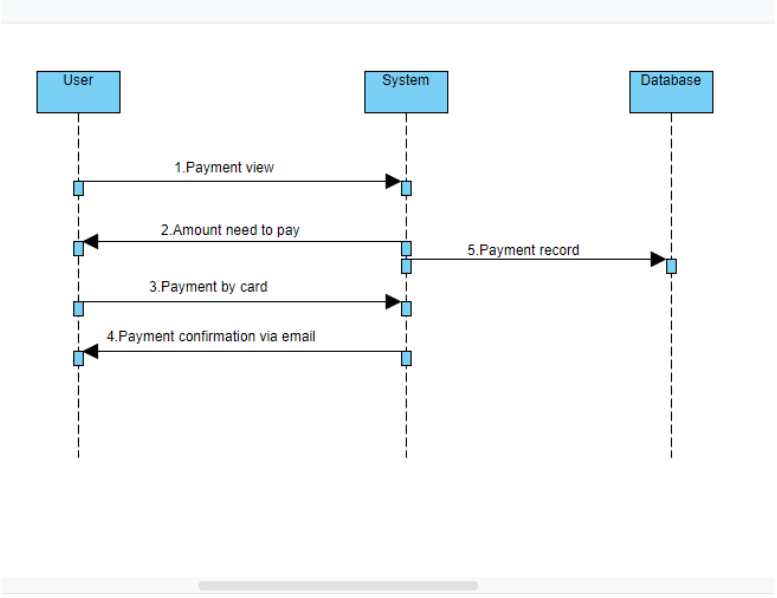
4.1.2 Parking Reservation Cancellation

In this diagram the user clicks on the cancel reservation button from the user dashboard and is brought to the cancel reservation page. The user then sends a request to cancel the reservation to the system and the system will first check if the reservation exists. The system will ask the user for confirmation to cancel the reservation. The user confirms the cancellation and the system will remove the reservation from the database, effectively cancelling it. The user can then see that the reservation has been cancelled.



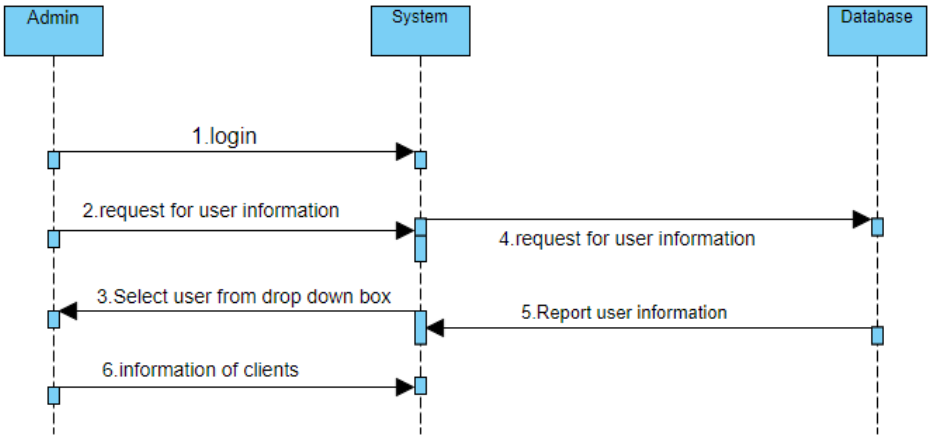
4.1.3 Parking Reservation Payment

The user accesses the parking payment page and is shown the amount to pay for the parking spot. User submits their payment information to the system and the system will store that payment information in the database. Afterwards, the system will issue a receipt to the user with the amount paid and the date.



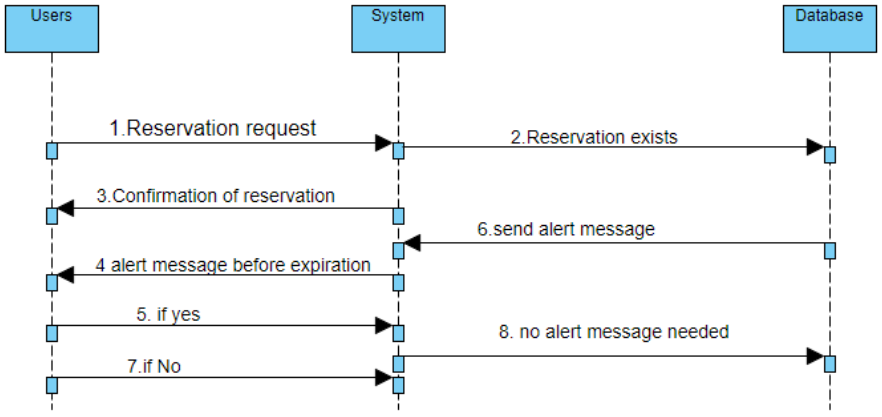
4.1.4 User information

The admin logs into the system and accesses the user information page. A drop-down menu will be populated with the list of users from the database. The admin will select the user whose information they wish to see from the drop-down menu. The system will pull the selected user's information from the database and then display it for the admin to see.



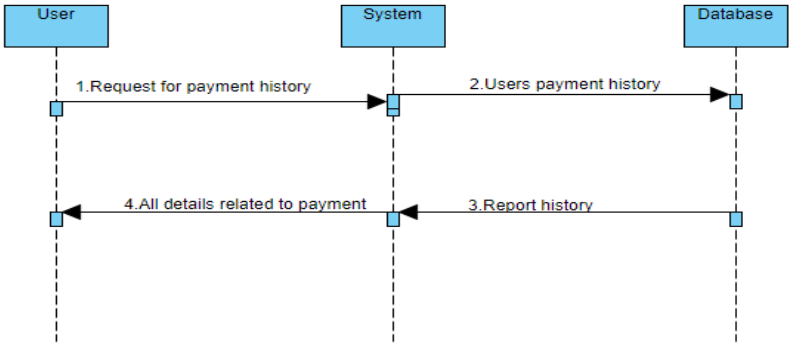
4.1.5 Notification of reservation expiration

After making a reservation the system will ask the user if they wish to receive a notification fifteen minutes before the expiration time. If the user says yes, the system will check the reservation's time from the database then issue the notification via the user's registered email fifteen minutes before expiration. If the user says no, then the system will not issue a notification.



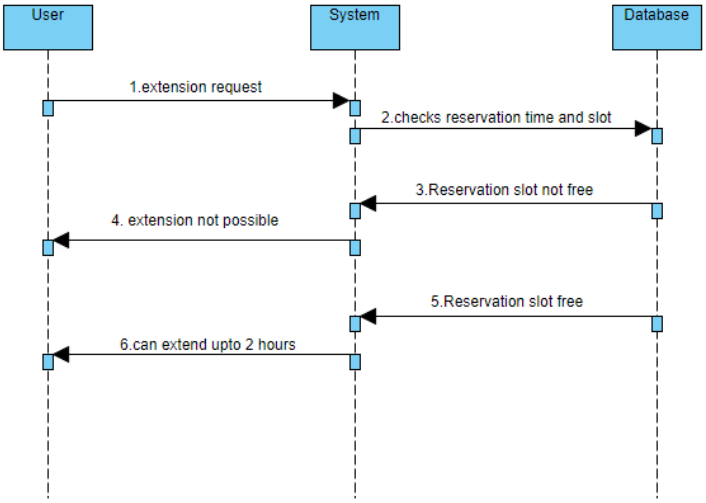
4.1.6 Payment History

The user clicks on the payment history button from the user dashboard page and is brought to the payment history page. While this is happening, the system pulls the user's payment history from the database and displays it for the user.



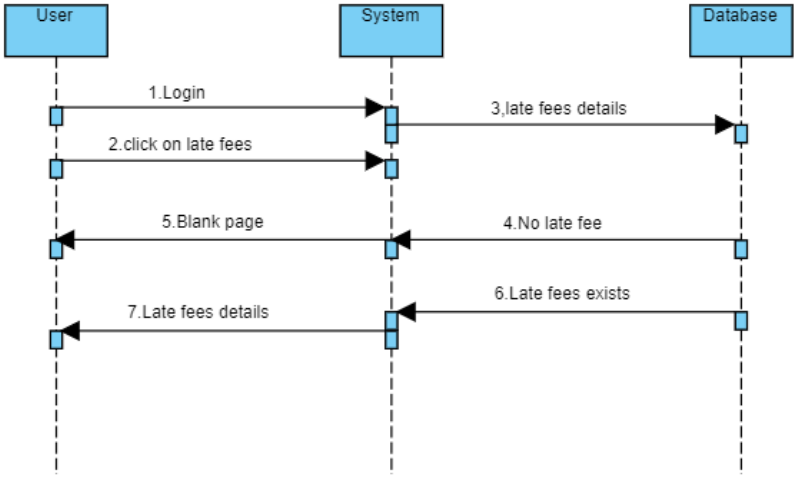
4.1.7 Extension of reservation

The user clicks on the extend reservation button from the user dashboard screen and is brought to the extend reservation page. The system will check the database to see if there are any reservations at the date/time the user wants. If there already is a reservation, the system will send the user a message saying that it is not possible to extend the reservation because there is already another reservation after their time. If there is no reservation, then the system will issue a message to the user saying that the time is free, and the user can then decide if they want to go through with the extension.



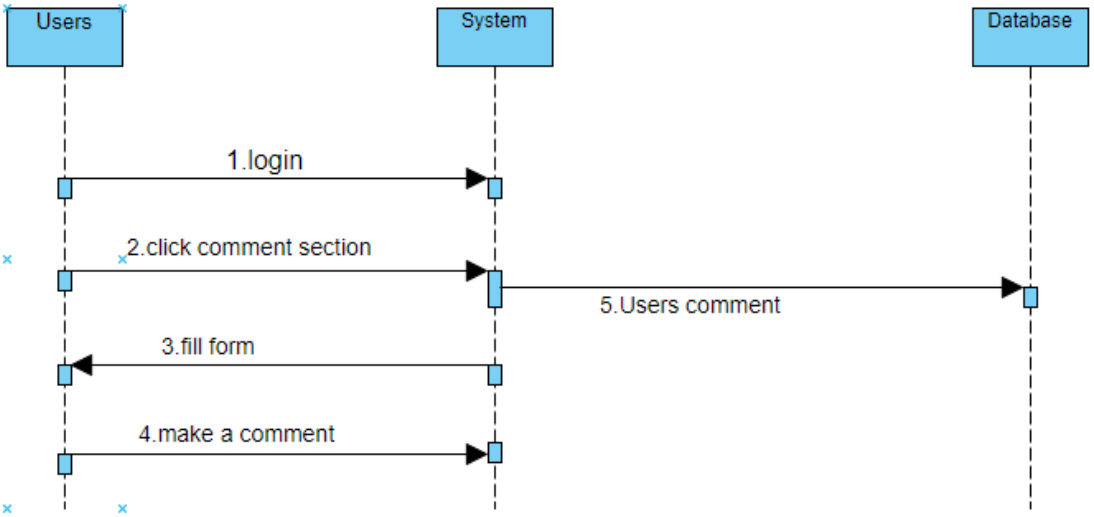
4.1.8 Payment of late fees

The user logins to the system and clicks on the pay late fees button on the user dashboard page. The user is brought to the late fees page and the system will then check the database to see if the user has late fees. If the user does not have any late fees, then the screen will show blank with a message saying no late fees. If the user does have late fees, then the screen will show the amount owed.



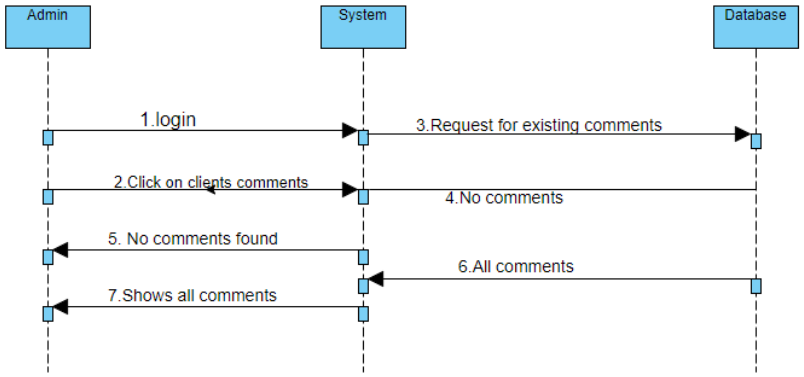
4.1.9 Submit Comments

The user logins to the system and clicks on the submit comment button on the user dashboard screen. The user then fills out the form with their information and the comment they wish to submit. The user submits the comment to the system and the system will store the comment in the database.



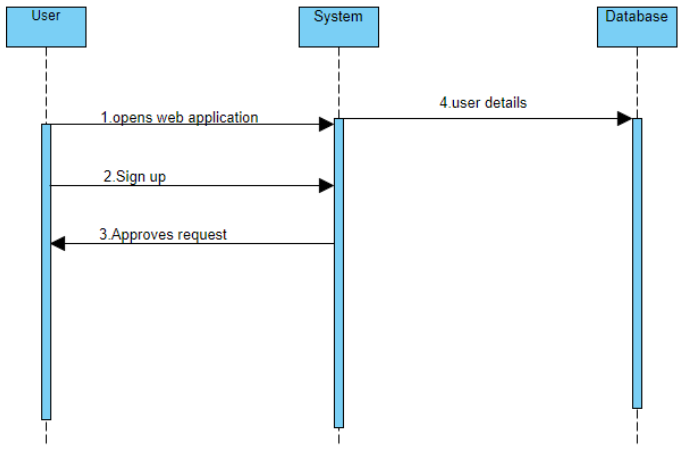
4.1.10 View comments

Admin logs into the system and clicks on the view comments button from the admin dashboard page. The system will check the database if there are any comments from users. If there are none, then the system will issue a message to the admin saying that no comments were found. If there are comments, the system will pull up the list of users who have submitted comments , along with the actual comments.



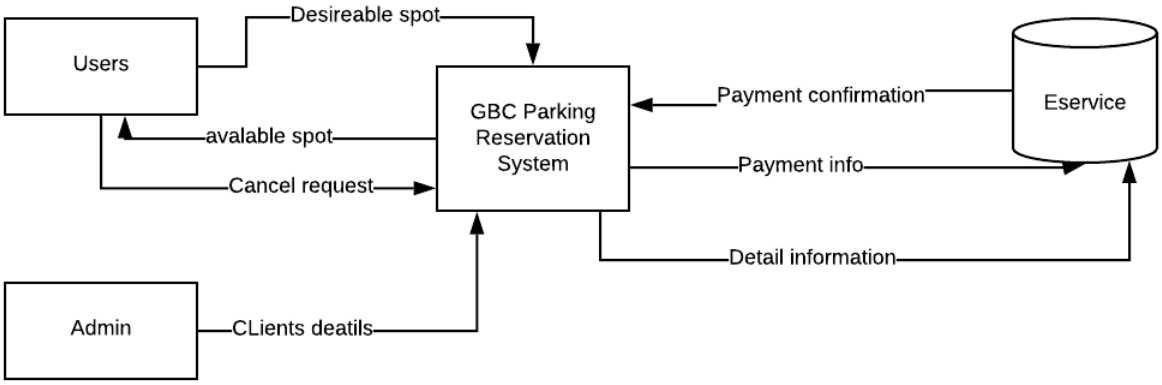
4.1.11 Sign up

The user accesses the application via the internet and reaches the login screen. The user clicks on the sign-up button and then fills out their information (GBC number, first name, last name, email). The user then clicks submit and the information is stored in the database. The user receives confirmation that the account was created.

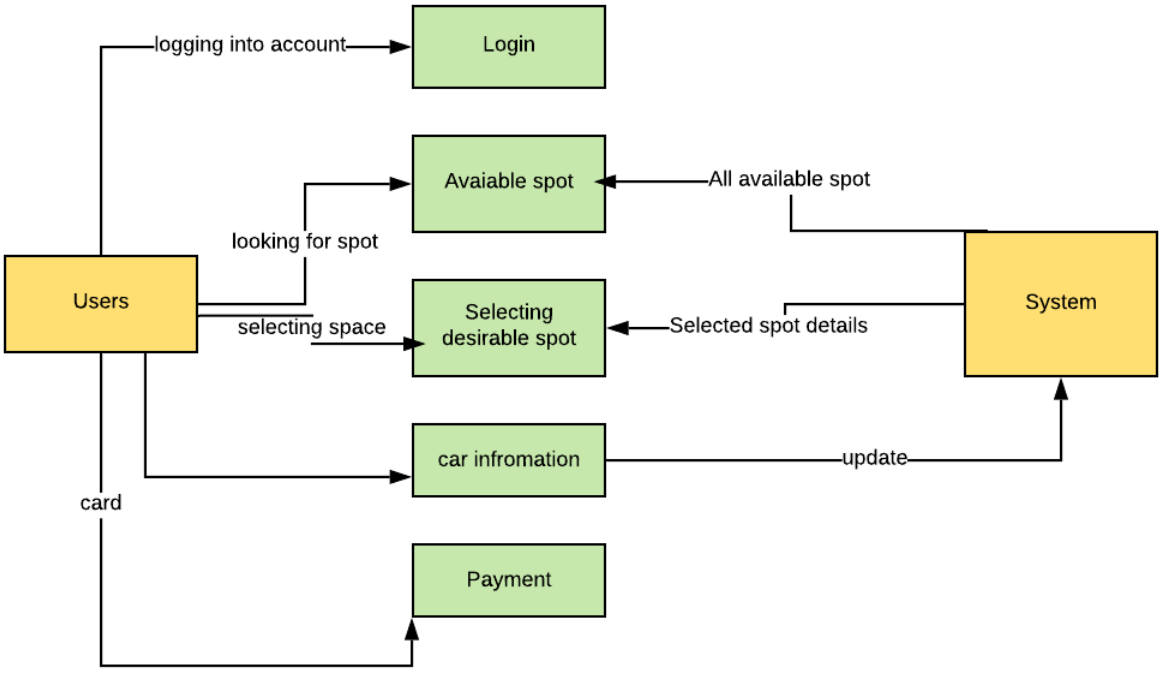


4.2 Data Flow Diagrams (DFD)

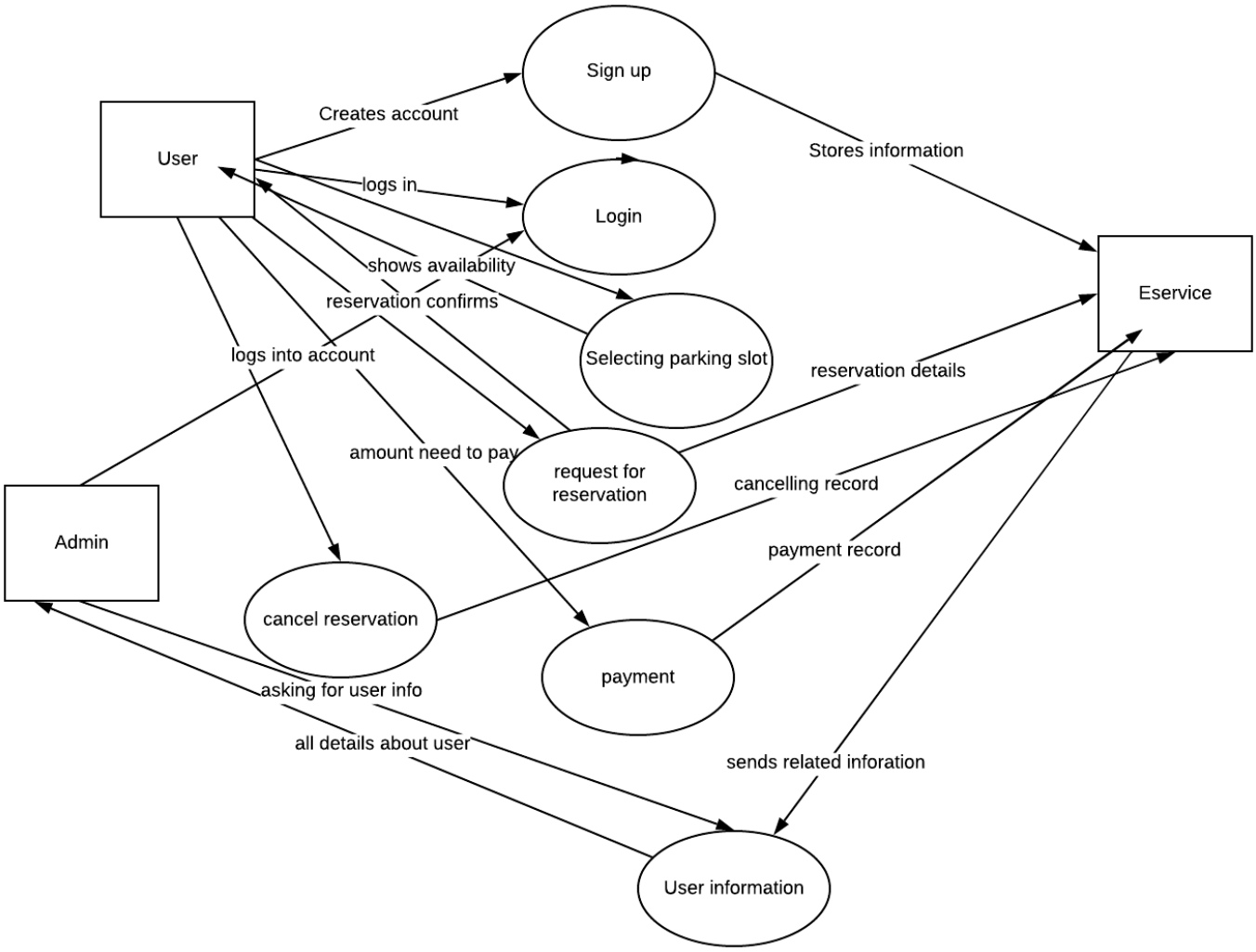
4.2.1 Level 0 DFD Diagram



4.2.2 Level 1DFD Diagram

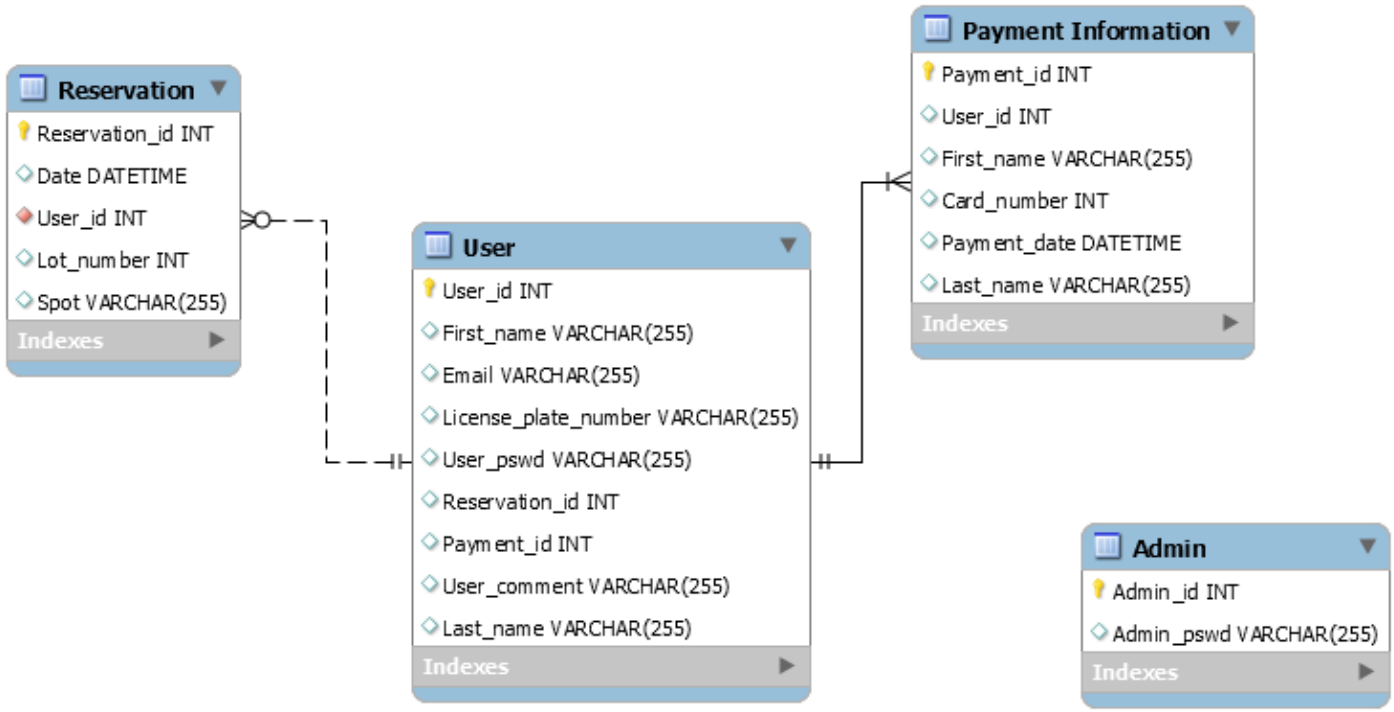


4.2.3 Level 2 DFD Diagram



4.3 Normalized Data Model Diagram

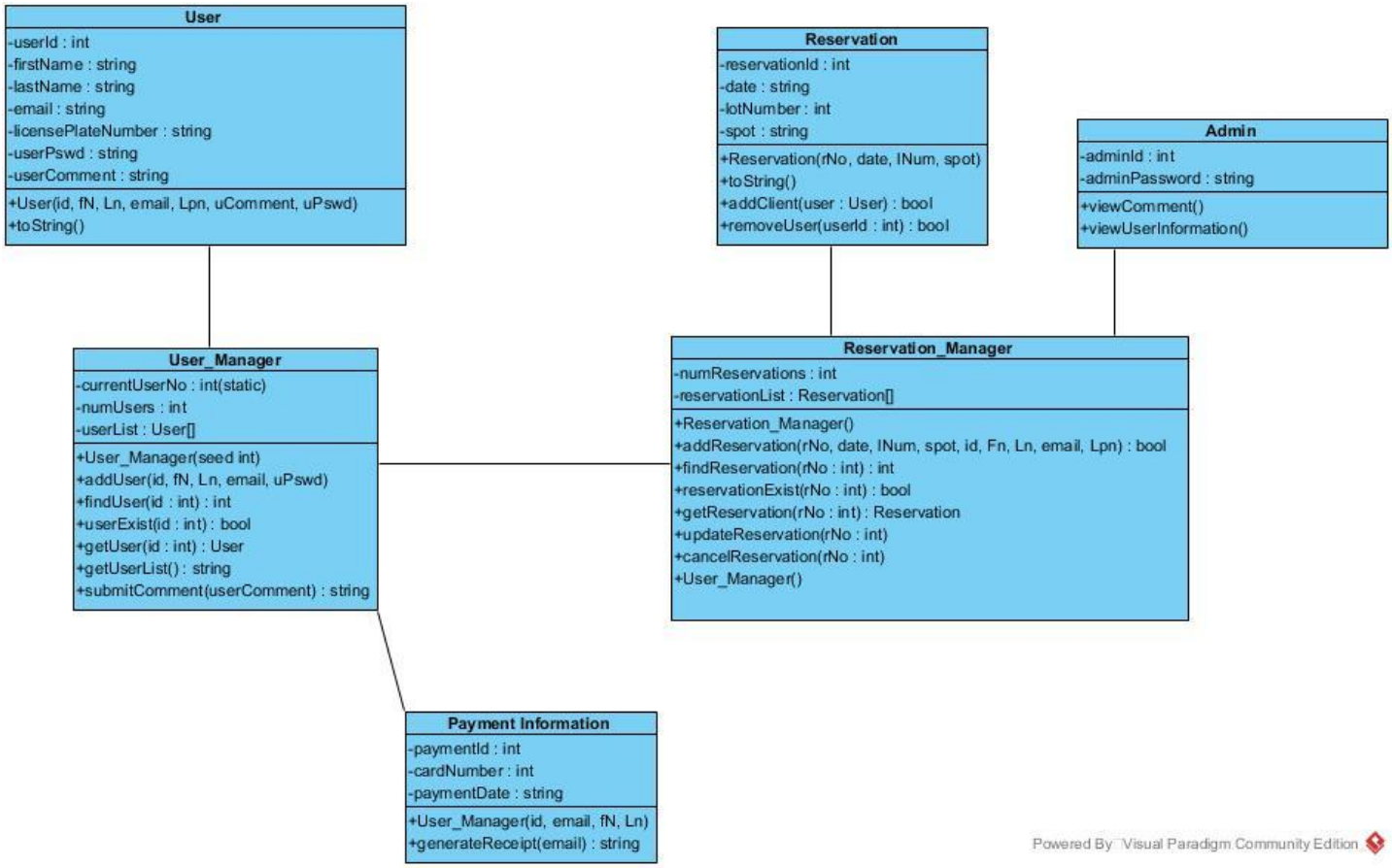
Below is the Normalized Data Model Diagram. This diagram shows the tables that will comprise the database and their relationships to each other. The user table is the largest table and will contain the User Id as the primary key, with Reservation Id and Payment Id as foreign keys. The reservation table contains the Reservation Id as the primary key and the User Id as a foreign key. The Payment Information table contains the Payment Id as the primary key and User Id as a foreign key. Lastly, the admin table contains the Admin Id as the primary key. The relationships between the tables are as follows: A user can choose to make a reservation and can have many reservations over their duration of using the system, but a reservation depends on having a user and can only have one user at a time. A user can optionally pay for their reservation online and can have many different payment methods, but the payment information requires there to be a user in order to exist. The admin table functions independently and does not rely on the other tables.



4.4 Unified Modelling Language Diagram (UML)

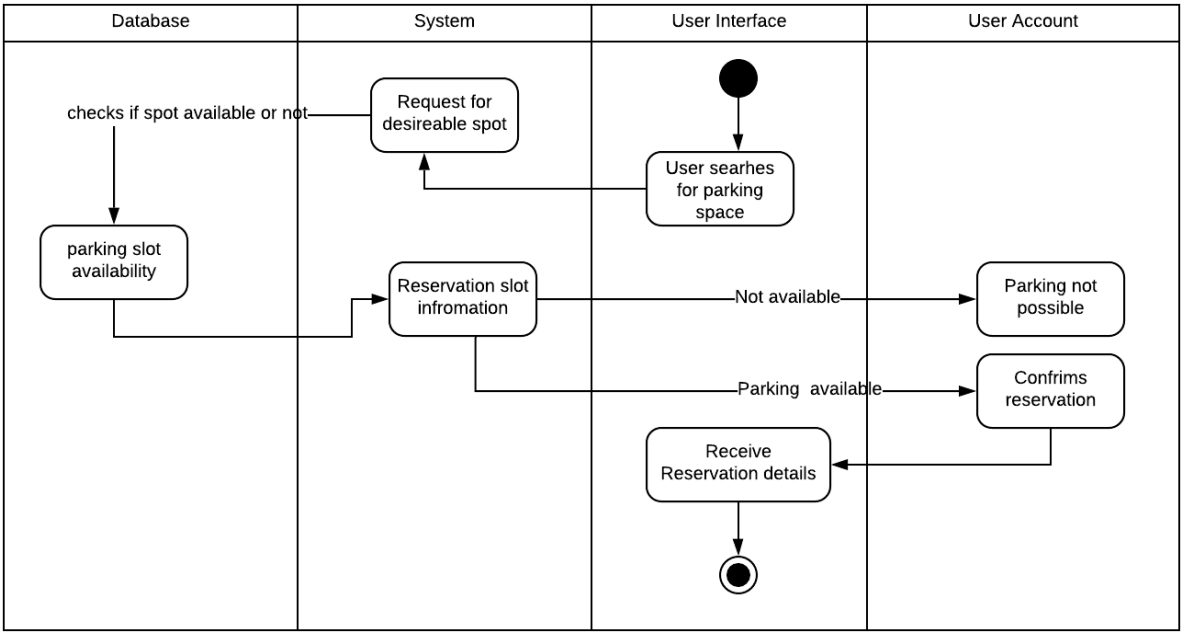
Below is the UML class diagram for this application. There are six total classes: User, User Manager, Reservation, Reservation Manager, Payment Information and Admin. The User Manager class is responsible for all functions that require information from the user table of the

database, such as signing up, submitting comments and displaying user information. The Reservation Manager class is responsible for all functions pertaining to a reservation, such as reserving a spot, extending a reservation, and cancelling a reservation. This class imports the User Manager class in order to access the attributes of the User class. The Payment Information class is responsible for storing the user's payment information and has one function, which is to generate a receipt to email to the user once they reserve a parking spot. This class imports the User Manager class to access the User Id, First Name, Last Name and Email attributes. The admin class contains information pertaining to the admin of the system, such as id and password. The admin contains two functions, which are view comments and view user list. Lastly, the User class contains the information pertaining to the user, such as name, etc.

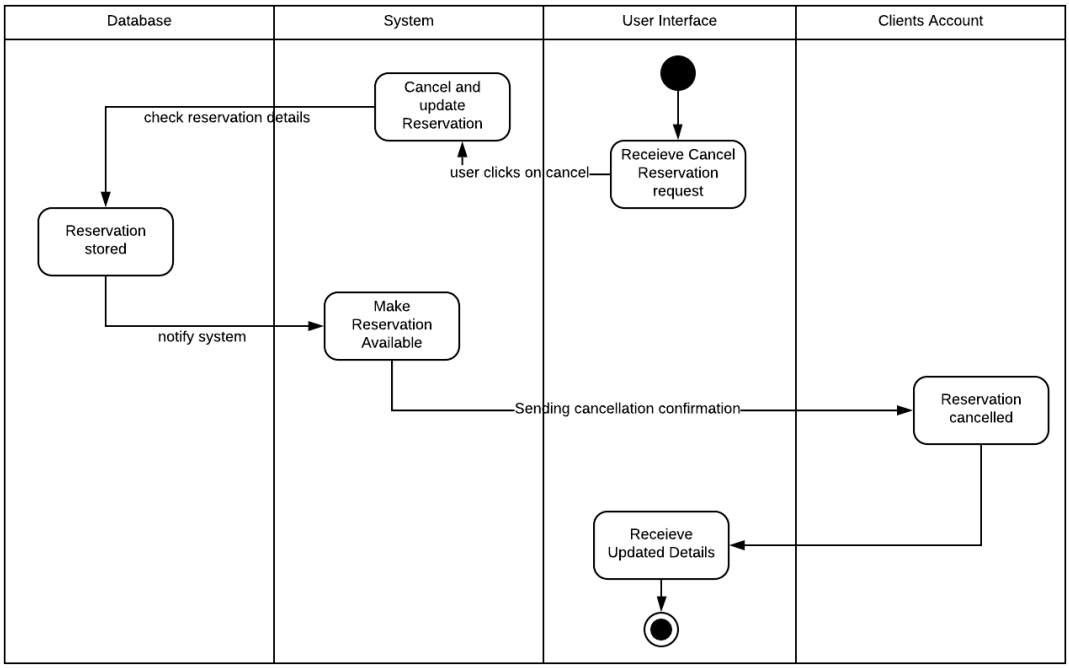


4.5 Activity Diagrams

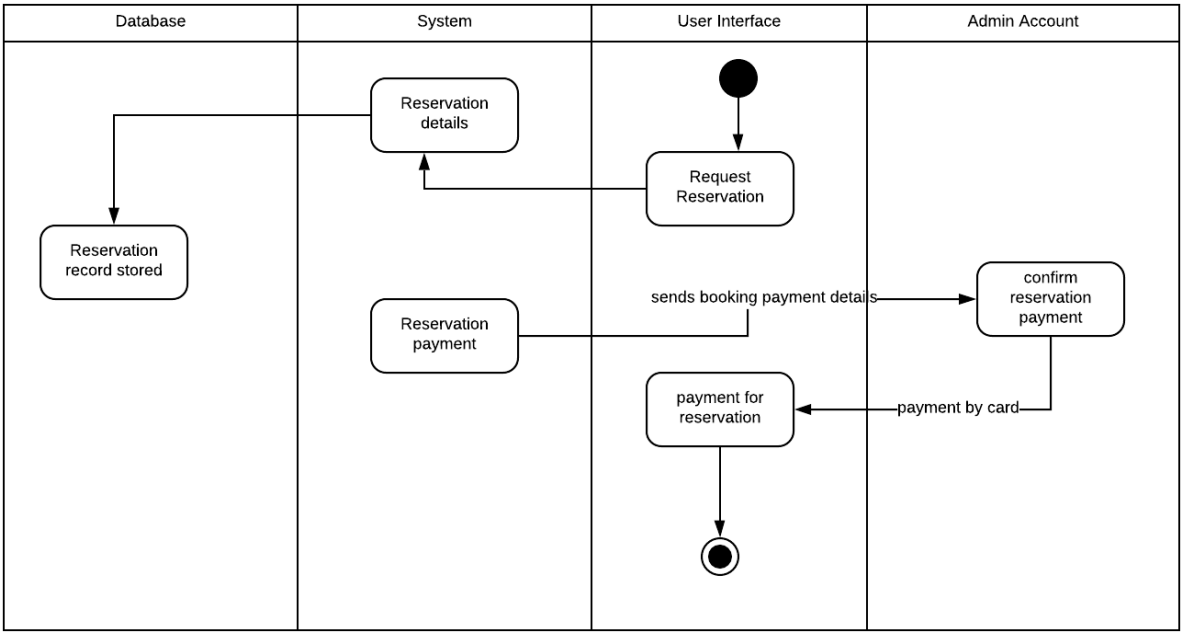
4.5.1 Parking Spot Selection



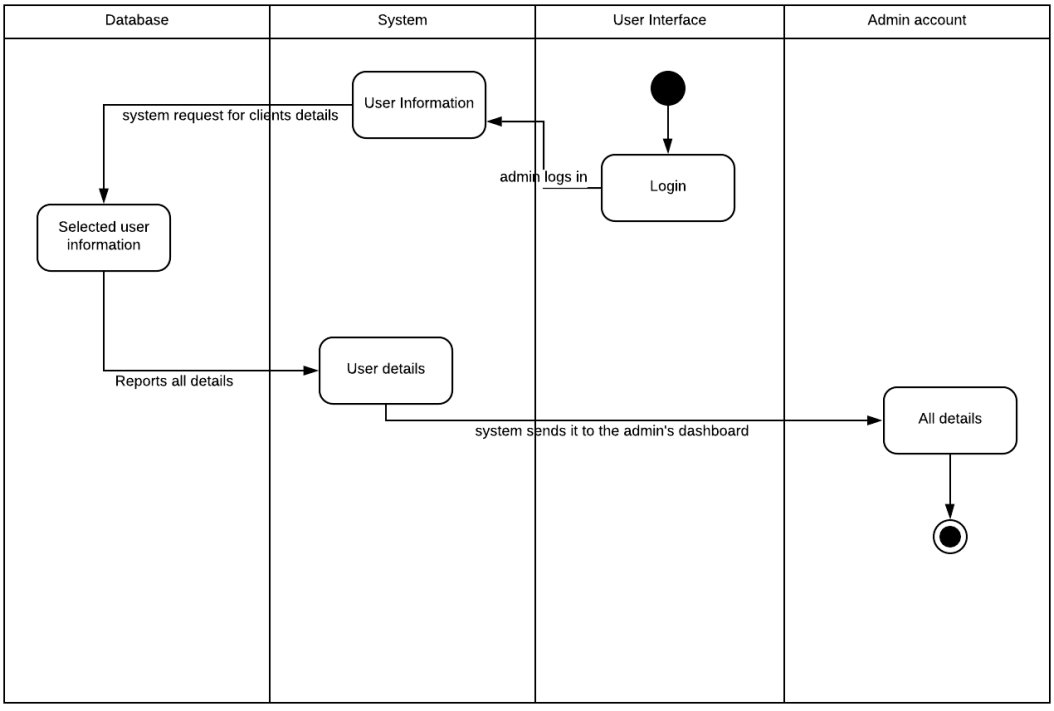
4.5.2 Parking Reservation Cancellation



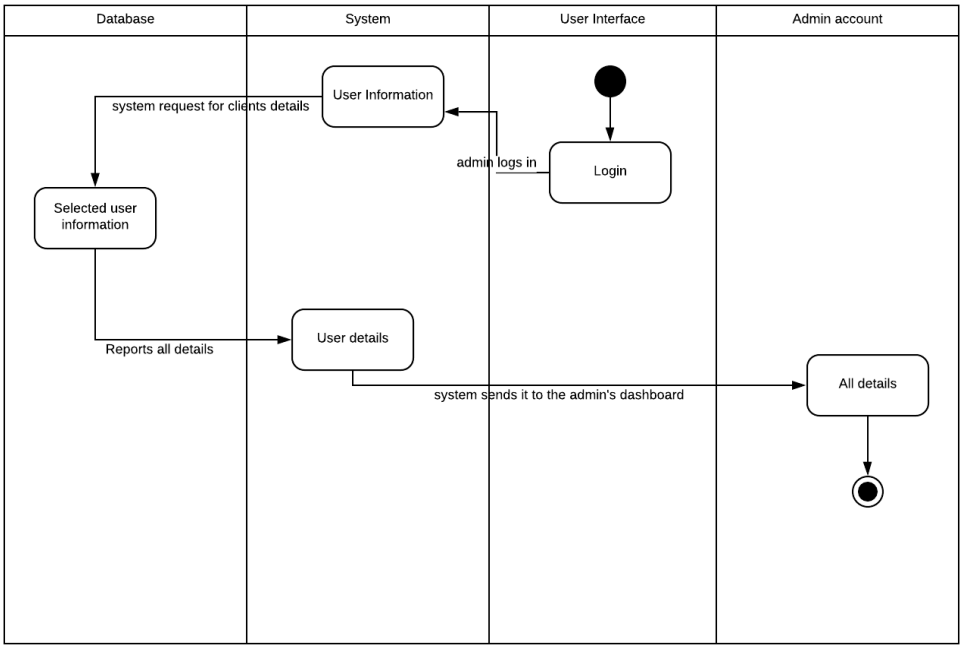
4.5.3 Parking Reservation Payment



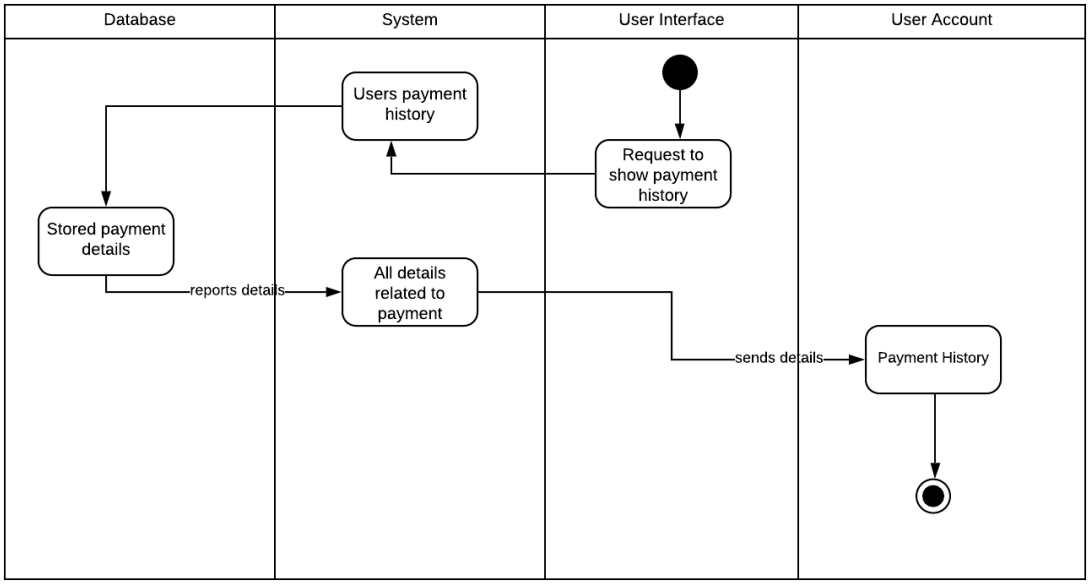
4.5.4 User Information



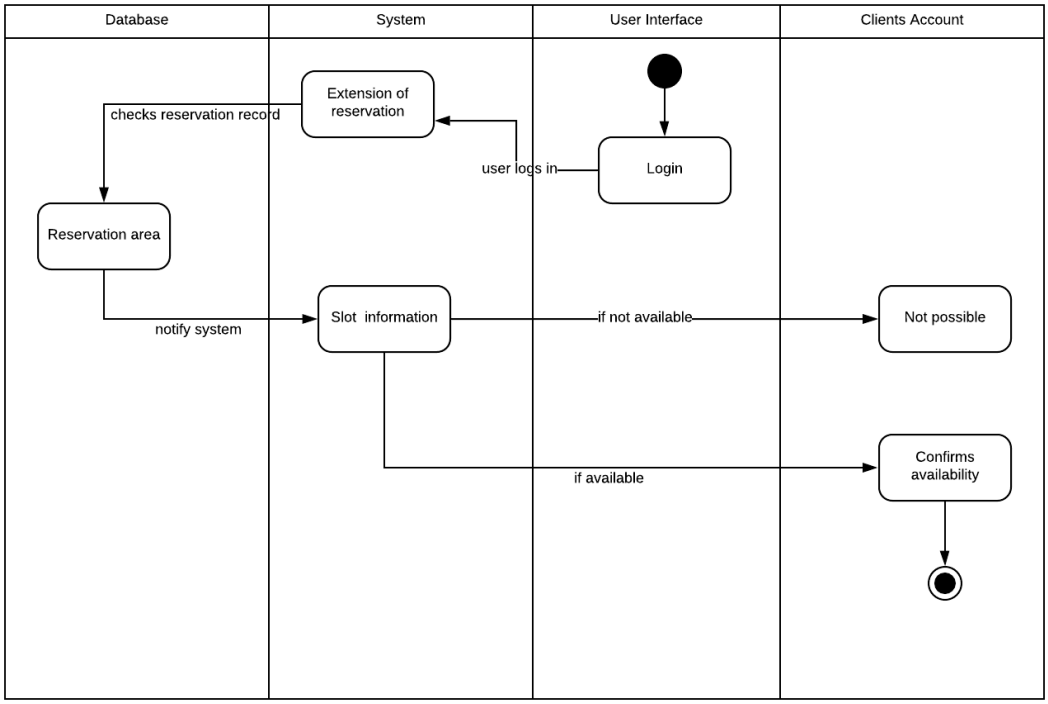
4.5.5 Notification of Reservation Expiration



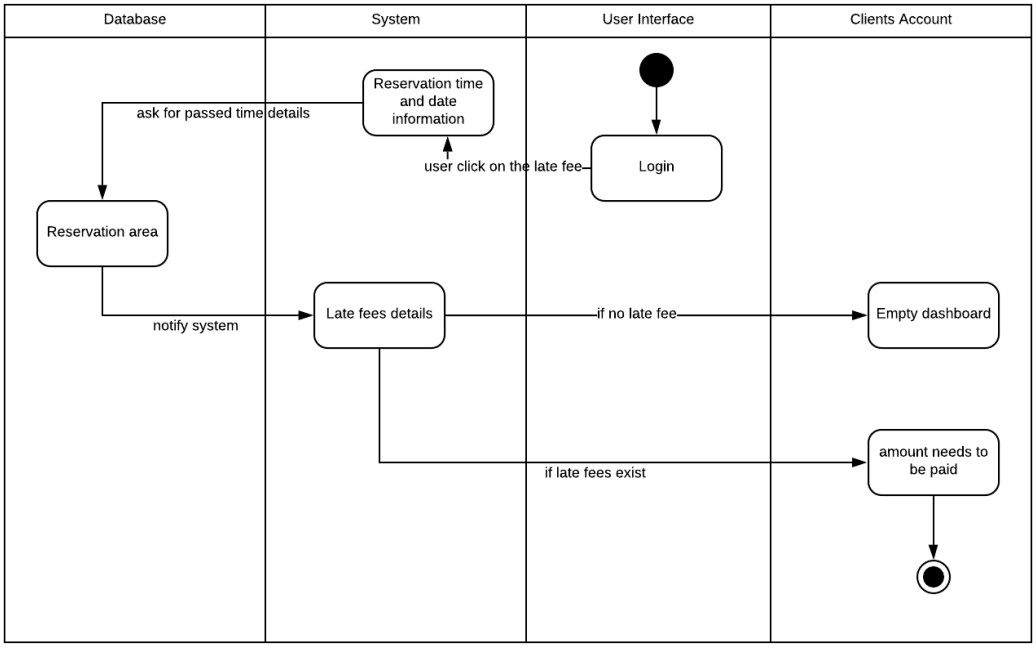
4.5.6 Payment History



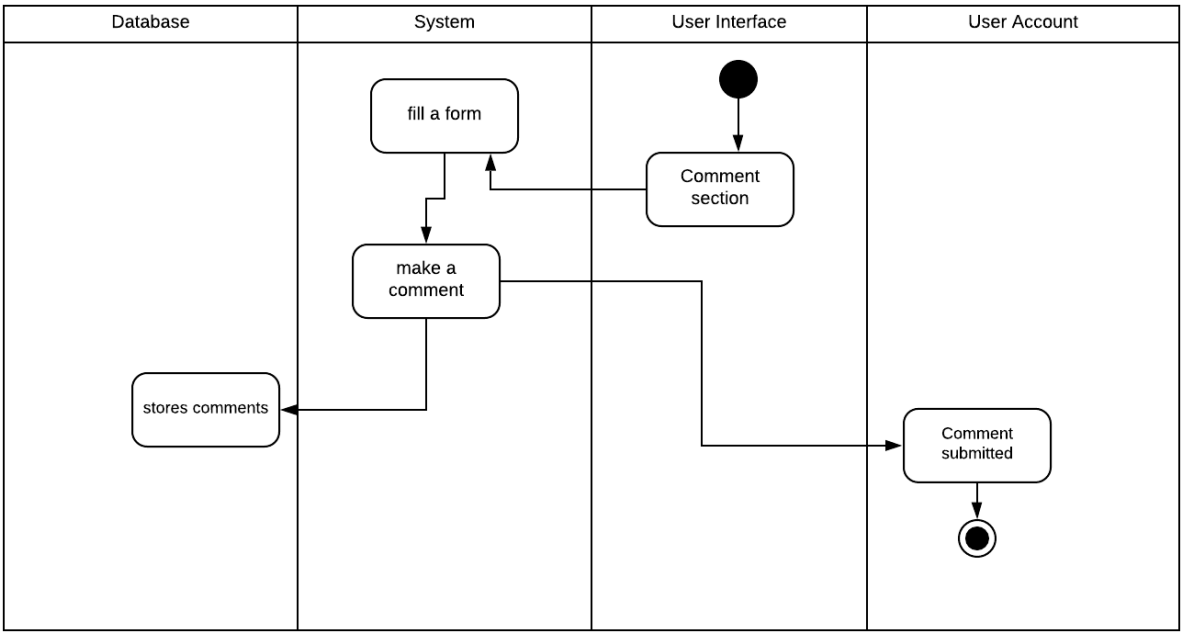
4.5.7 Extension of Reservation



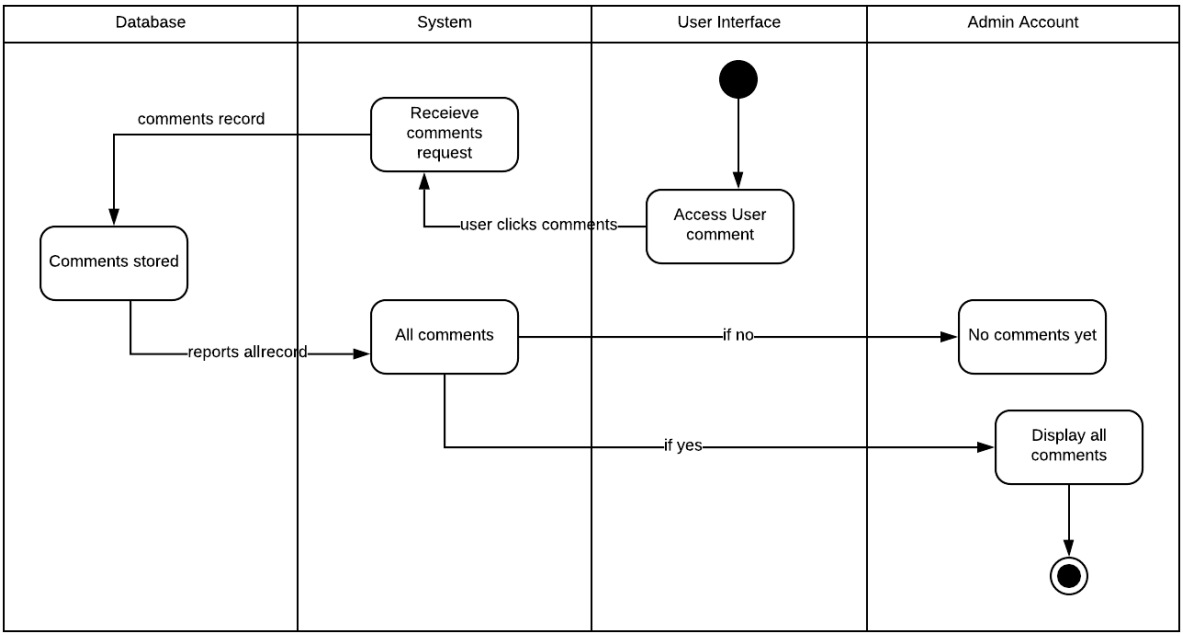
4.5.8 Payment of Late Fees



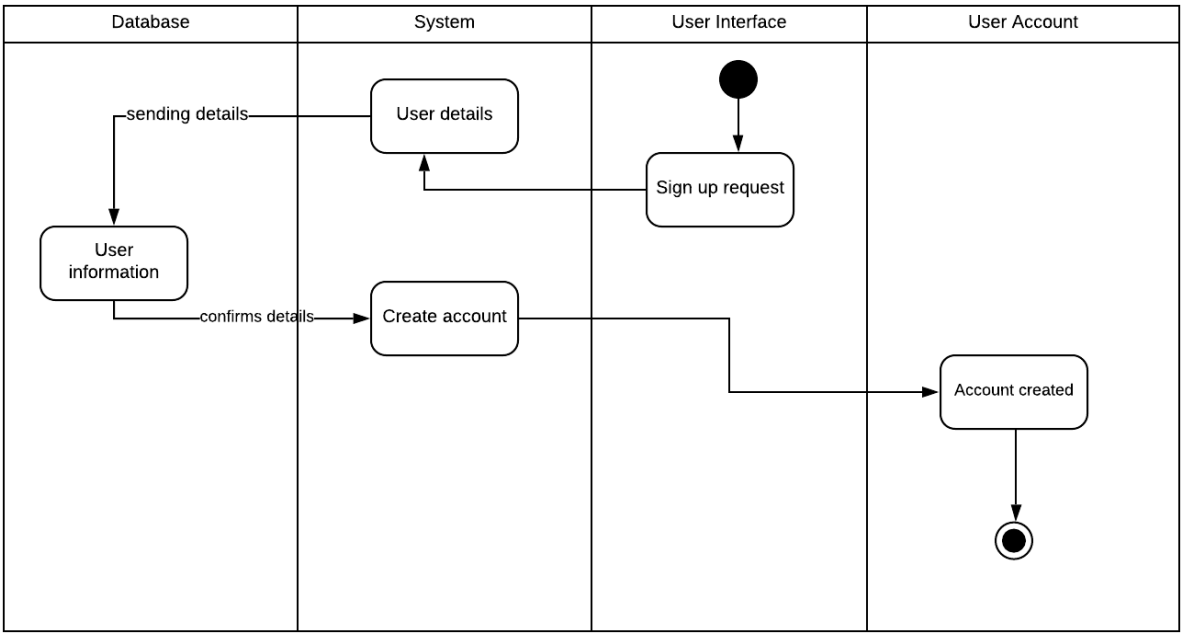
4.5.9 Submit Comments



4.5.10 View Comments



4.5.11 Create Account



5. Change Management Process

If changes need to be made to this document, the lead developer will convene a meeting with the other developers to discuss the proposed changes. The team will discuss and eventually reach a consensus, then the changes will be implemented.