CSN 291

PROJECT REPORT

PARKIFY

8th November 2022

Submitted by-

Group 2-

- Rajat Raj Singh. *Email* - r_rsingh@cs.iitr.ac.in Mob. No.- 8218516522
  Enrollment no. - 21114079
  Contribution- Basic Framework of the application, Page Linking, Database Management, Shared Preferences, Use Case Diagram and Flowchart

- Priyanshu Behera. *Email* - p_behera@cs.iitr.ac.in Mob. No.- 9340371238
  Enrollment no. - 21114077
  Contribution- UI Design and Sequence Diagrams

- Priyansh Mawal. *Email* - p_mawal@cs.iitr.ac.in Mob. No.- 6378146484
  Enrollment no. - 21114076
  Contribution- UI Design and Slides

- Piyush Arya. *Email* - p_arya@cs.iitr.ac.in Mob. No.- 9116916870
  Enrollment no. - 21114074
  Contribution- Basic Framework of the application, Database Management, Shared Preferences, Class Diagram

- Pranavdeep Singh. *Email* - p_singh2@cs.iitr.ac.in Mob. No.- 8920582347
  Enrollment no. - 21119036
  Contribution- Basic Framework of the application, Page Linking, Database Management, Shared Preferences, Use Case Diagram and Flowchart

- Atharv Chhabra. *Email* - a_chhabra@cs.iitr.ac.in Mob. No.- 6260888533
  Enrollment no. - 21118025
  Contribution- UI Design, Nearest Slot Algorithm, Project Report

➢ Problem Statement:

With the continuously increasing global population, it's no surprise that the parking process in large parking spaces has become very cumbersome and inefficient. People keep searching for vacant parking spots in large places such as malls, only to have wasted their valuable time for something which could have been dealt with very easily, with just the knowledge of the nearest vacant parking spot. We've tried to design an application to deal with the same and make the overall parking process easier and more efficient, while maintaining a simplistic user interface.

➢ Decomposition into Sub-Problems:

The problem that we aimed at solving was to provide the user with the knowledge of the nearest vacant parking spot. We decomposed this problem into the following sub-problems:

- Information needed from the user to maintain track of the parking spots' availability.
- The problem of storing this data somewhere in order for the necessary program to determine the nearest vacant parking spot.
- Algorithm to determine the nearest vacant parking spot and update the parking spots' data as necessary.

- Calculation of the net fair for parking the vehicle for a given duration.
- Designing the app user interface.

➢ <u>Requirements Analysis:</u>

- Functional Requirements (in the form of user stories):
  As a user, I'd like to be able to enter the required data in the app quickly and without hassle. In case I make a mistake while entering, I should be presented with an opportunity to re-enter the correct data.
  Once the data is entered, I would like to be presented with the details of the nearest vacant parking spot quickly. On exit, I'd like to enter the relevant details quickly and be presented with the opportunity to correct myself in case of an error. Thereafter, I would like to know the amount that needs to be paid without any delay.

- Non-functional Requirements:
  The nearest vacant spot computation algorithm should be fast. The design and interface should be modifiable and updatable. The application should be compatible with the latest versions of android and shouldn't lag whatsoever. The computation algorithm should give the correct answer i.e. be reliable.
- User-interface Requirements:
  It should avoid unnecessary complexity and be as simple and easy to use as possible. It should redirect the user to the

appropriate screen in case the user makes a mistake in entering their data. It should be lightweight to avoid crashes and errors, while having an intuitive and attractive user interface.

➢ <u>Use Cases:</u>

There are 2 use cases:

❖ Use Case 1 (Entry):

Scenario 1: Mainline Sequence:

1. User: Press Entry button.
2. System: Asks for User details and PIN.
3. User: Enters details and sets PIN.
4. System: Stores this information and shows the allotted slot.

❖ Use Case 2 (Exit):

Scenario 1: Mainline sequence:

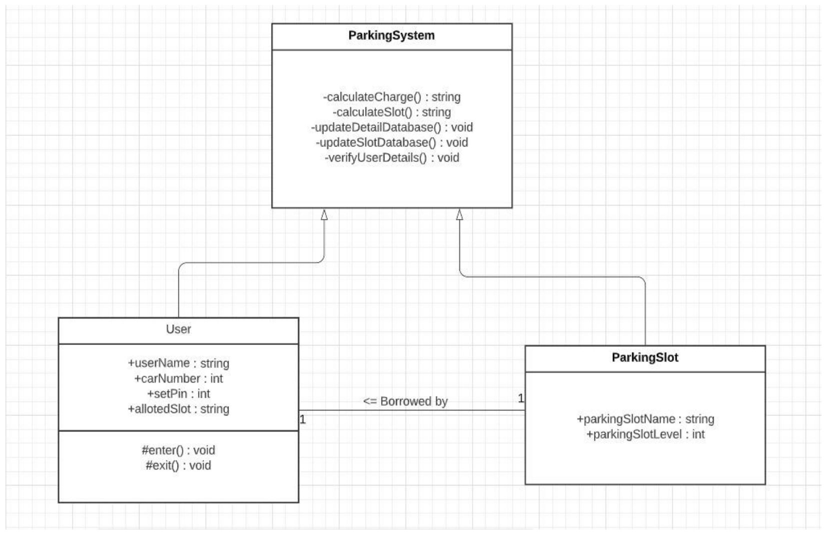
1. User: Press Exit button.
2. System: Asks for details of the user.
3. User: Enters the details.
4. System: Updates databases, deletes the record of those details, calculate the duration and display the charge.

Scenario 2: At the end of step 3:

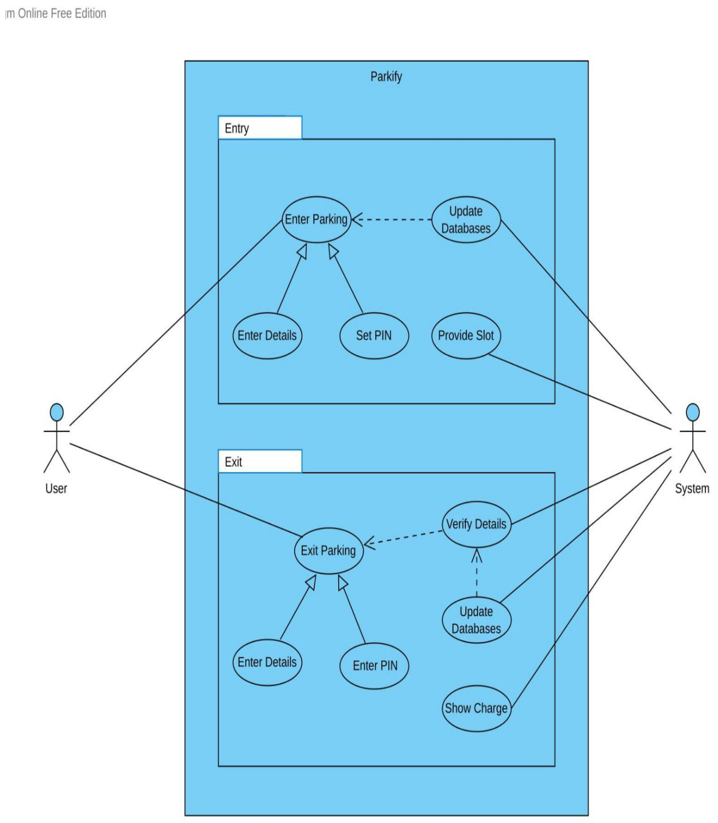
3) System: Returns error message of the details not matching to any record in
current database.

➢ <u>UML Diagrams:</u>

Class Diagram:

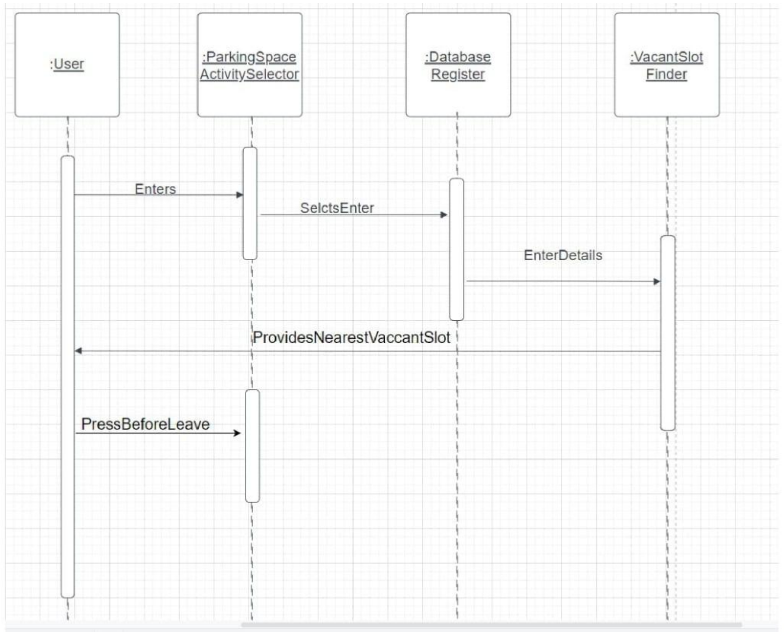


Use Case Diagram:

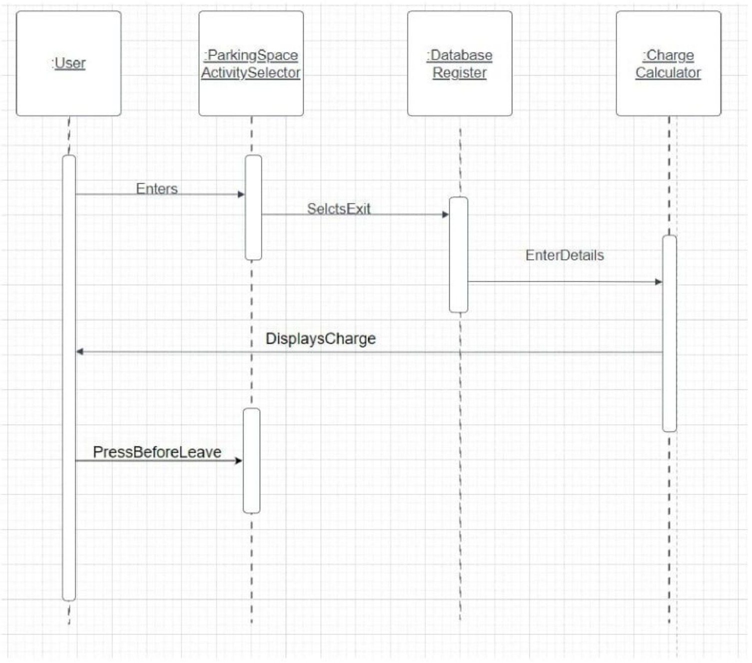


Sequence Diagram:

For entry:



For Exit:



➢ User Experience:

The application presents the user with a very simplistic and intuitive user interface which is easy to use and understand. We didn't complicate the design unnecessarily to maintain the lightness of the application and prevent crashes and bugs. It's extremely simple for the user to understand and operate the application with the given few choices of self-explanatory buttons. They're presented with the opportunity to re-enter their data in case of an error.

➢ Time dependencies:

The time dependencies in our application are for the specific end users who'd utilize the parking spaces. They would be charged $50 per 10 sec (for demonstration purposes) displayed on the main screen of the application that they view while entering their data. Correspondingly, they would be charged the appropriate amount on exit.

➢ Hardware Requirements:
Android Version 5.1 and above

➢ Algorithm used for computation of nearest vacant parking spot:

We went with a total of 4 floors for the parking system, with 100 parking spots in each floor subdivided into 10 rows and 10 columns. The parking spot [row, column] is represented as XC, where X is the row number and C is the character in the alphabet at the number "column". E.x. Parking Spot [3,5] is represented as 3E. The program to calculate the nearest vacant parking spot would receive the data of the parking spots of all floors in the form of a string of 400 characters, with each 100 consecutive characters representing the data relevant to a given floor. Thereafter, within each row, 10 consecutive characters describe the status of parking spots in a given row and 10 columns.

It's important to note that an assumption was made regarding the nearness of parking spots among floors. It's assumed that the spot [1,1] on the next floor is further than the spot [10,10] on the corresponding floor.

The least floor with the cell [10,10] as vacant was processed. A key observation is that the distance of the spots from the entry point can be simplified to a formula: Spot [i,j] is at a distance of i+j+1 from the entry point. As a result, the nearest parking spot on a floor is just a vacant spot with the least value of i+j+1 which can be found in O(n). The relevant parking spot with the corresponding floor was then returned to the database management system and thereafter displayed to the user.

➢ <u>Database of the application:</u>

We used 2 different tables in our app-

- Slots table- it stores the present state of the slots in our parking area. It stores boolean values 0 and 1 where 0 represents vacant and 1 represents allotted. When the app starts for the first time all the slot values are initialized as 0. After an entry, the nearest parking slot is alloted and the state for that slot is updated to 1. After an exit, we reset that slot state to 0.

- User details table- It stores the details of the users who enter the parking area along with the allocated slot and time of entry. At exit, the user details are deleted from the table.

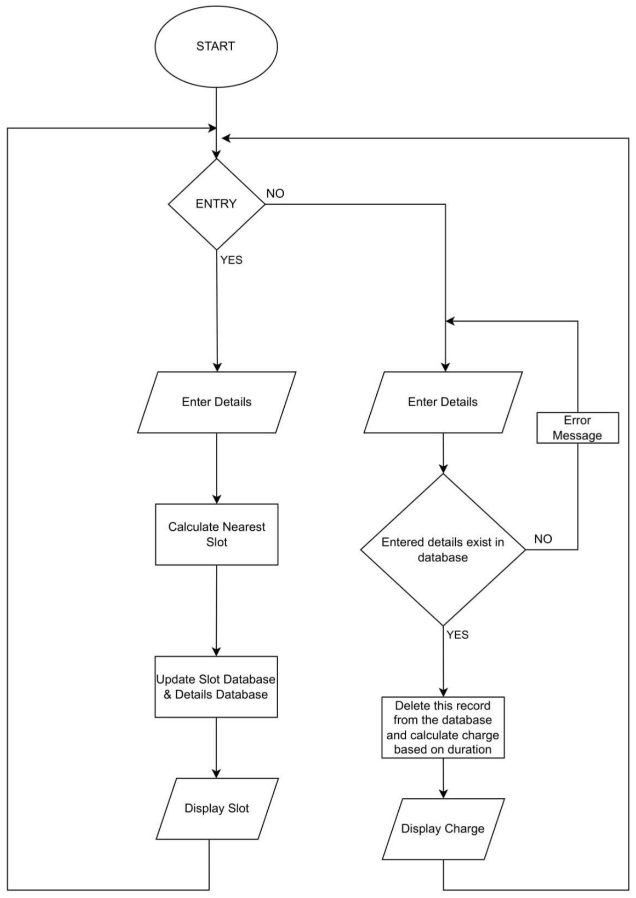
➢ <u>Shared Preferences and execution flow of the application:</u>

Shared preference is a way to store small data items whose values are retained even when the application is closed and can be retrieved/accessed in the app.
We have used shared preferences to store 2 values:

- Counter: It is zero for the first time the app is opened and is 1 for all other times. This is required as we create and initialize the databases only when the counter is zero.
- Time: It is an integer which is incremented every second, it is used to calculate the duration of parking for a particular user.

An Intent is a messaging object which we have used in Parkify to switch from one activity(one screen in app) to another activity. It can also pass data across activities. The screen at the start(Welcome screen) is linked to the main screen(screen 1) where the user will select Entry/Exit. No data is passed in this phase. If the user selects “Entry” then the app changes to screen 2 where the user will enter personal details. Nearest slot calculation is done and passed by the intent while going from screen 2 to screen 3. The nearest slot will be displayed on screen 3. On screen 3 a button will redirect us to the main screen. If the user selects “Exit” then the screen changes to screen 4 where the user is asked to enter the same details which were entered upon “Entry”. Parking charge will be calculated and passed by the intent while going from screen 4 to screen 5. The charge will be displayed on screen 5. A button on screen 5 will again redirect us to the main screen.

Execution order explained using a diagram/flow-chart:


START
ENTRY
NO
YES
Enter Details
Calculate Nearest Slot
Update Slot Database & Details Database
Display Slot
Enter Details
Error Message
Entered details exist in database
NO
YES
Delete this record from the database and calculate charge based on duration
Display Charge

➢ References Used:

https://www.freepik.com/free-vector/isometric-multistorey-car-park-with-parked-vehicles-vacant-lots-3d-vector-illustration_23182505.htm

https://www.freepik.com/free-vector/underground-car-parking-garage-with-vehicle_21388606.htm#page=2&query=parking%20lot&position=49&from_view=keyword

https://www.freepik.com/free-vector/parking-concept-illustration_16482945.htm#query=parking%20garage&position=4&from_view=search&track=sph

https://www.freepik.com/free-vector/dollar_2900482.htm#query=money%20vector&position=48&from_view=search&track=sph

https://www.freepik.com/free-vector/sport-car-cartoon-vector-icon-illustration-transportation-object-icon-concept-isolated-premium-vector-flat-cartoon-style_21186830.htm#page=3&query=car%20vectors&position=6&from_view=search&track=sph

https://stackoverflow.com/questions/36236181/how-to-remove-title-bar-from-the-android-activity