Vaibhav Nanda
Kaushal Tiwari
Sushant Malik

PARKING GUIDANCE SYSTEM

PROJECT REPORT

Guide: Ms. **Rachna Sethi**

ACKNOWLEDGEMENT

Exchange of ideas generates the new object to work in a better way. Whenever a person is helped and co-operated by others, his heart is bound to pay gratitude and obligation to them. To develop a project is not a one man show. It is essentially a collective work, where every step is taken with all precautions and care. Therefore, our first duty is to thank all those who took pain in completing this project.

Firstly, we thank Ms.Rachna Sethi, who gave us inspiration and guidance to work our way towards this field and gave us her precious time and attention whenever needed. Thanks may be a matter of mere formality but with us it is an expression of heartfelt gratitude towards our project supervisor. We are highly indebted for her advice, perceptive guidance and constant encouragement at every point of time during this study. The timely and persistent advices and assistances offered are greatly acknowledged.

We would also like to thank Mr. Arjun Sethi, Parking Manager, DLF Emporio, for his precious guidance and review of our system in comparison to the already established parking guidance system at Emporio.

CERTIFICATE

This is to certify that Kaushal Tiwari, Sushant Malik and Vaibhav Nanda, students of B.Sc.(Hons) Computer Science Semester IV have submitted the project entitled “Parking Guidance System” for the partial fulfilment of the requirements of B.Sc.(Hons) Computer Science.

It embodies the work done by them during semester IV of their course under the due supervision of Ms.Rachna Sethi

Date

Ms.Rachna Sethi

INDEX

INTRODUCTION

OVERVIEW

Personal vehicle usage has increased with increasing population and with the economic boom in India which creates serious problem of parking place at market areas, shopping malls and other public places. As per a survey, it is roughly estimated that out of 8760 hours in year, a car runs for an average of only 400 hours leaving 8360 hours in parked condition. Increasing concentration of human activity on limited land both in terms of residential activity and commercial activity causes paucity of space and the need for newer parking spaces puts more pressure on already overcrowded land. Every car owner would wish to park the car as closely as possible to his destination so as to minimize his walking distance leading to congestion of On-street spaces in official neighbourhoods and may give rise to inappropriate parking area in office and shopping mall complexes during the peak time of official transactions.

The demand also leads to economic, social and environmental losses and with increase in population the problem becomes more critical. As such parking spaces optimization and control or proper usage of available parking land has become a real challenge for city transport planners and traffic authority. By comparing various parking guidance systems and the characteristics required by a parking guidance system, we find that they are viable and suitable for Indian environment. These characteristics form the basis for designing Parking Guidance System for cinema theatres, malls, hotels and offices in India. These parking systems requires proper management with context to traffic management, allotted parking lot direction indication, vehicle identity, driver identity, vehicle safety, parking charges as per time duration, maintain data log, database and ledger management etc… Embedded systems can provide quality efficient, cost effective solutions to manage parking requirements.

While surveying the city for viable parking spots, we found certain areas in Delhi which have been very effectively used for parking by the local residents. One that caught our eye was the area beneath the rajouri garden flyover. Though the parking is not properly made and most people don't know that it is an available parking space, the idea was itself unique and a really space efficient one. Now, if the same concept with full automation was applied to all such viable under-flyover spaces, we might be able to solve the parking problem of the city while generating huge revenues for the government as well.

OBJECTIVE

In the modern world paucity of space has become a very big problem. In the era of miniaturization it has become a very crucial necessity to avoid the wastage of space in any manner. In space where more than 100 cars need to be parked, this is a very difficult task to accomplish.

In order to overcome existing systems problems, this new system is developed. The objectives of this parking guidance system:

- To ensures proper planning and safety of vehicle and the convenience of easy and ordered vehicle parking.
- To make the entire process of vehicle parking hassle-free and affordable with automated billing system giving the vehicle owner a respite from the constant tension of car parking.
- Generation of a user friendly interface.
- Generation of a systematic, easy to access and secure database.
- Generation of a system for guiding vehicles to the appropriate parking spot.
- To provide an administrator-view that will allow authorized employees to view and administer vehicle database, parking status etc.

PROBLEM STATEMENT

As per a survey, it is roughly estimated that out of 8760 hours in year, a car runs for an average of only 400 hours leaving 8360 hours in parked condition.

Increasing concentration of human activity on limited land both in terms of residential activity and commercial activity causes paucity of space and the need for newer parking spaces puts more pressure on already overcrowded land. In the modern world paucity of space has become a very big problem. In the era of miniaturization it has become a very crucial necessity to avoid the wastage of space in any manner.

As such parking spaces optimization and control or proper usage of available parking land has become a real challenge for city transport planners and traffic authority. By comparing various parking guidance systems and the characteristics required by a parking guidance system, we find that they are viable and suitable for Indian environment.

A parking guidance system not only saves a lot of time and nerves in finding a free parking space, it also reduces the traffic searching for parking spaces in a car park and therefore reduces the emission of pollutants and noise. The capacity utilization and the acceptance of the car park are increased.

Our vision is of a smarter, modern India which is free from unordered and inefficient parking space. Urban areas, where the parking space is in acute shortage, will be our arena of operation.

SCOPE OF SOFTWARE

The function of the Parking Guidance System is to control and supervise the entries and exits into and out of the parking lot. The system allows or rejects entries into the parking garage depending on the number of available parking space in the lot.

It obtains its optimality by applying modern technological methodologies and providing maximum number of parking spots in the limited area. It ensures proper planning and safety of vehicle and the convenience of easy and ordered vehicle parking.

It provides an administrator-view that will allow authorized employees to view and administer vehicle database, parking status, check identity of vehicle owner etc.

It makes the entire process of vehicle parking hassle-free and affordable with automated billing system giving the vehicle owner a respite from the constant tension of car parking. It has a system for guiding vehicles to the appropriate parking spot.

While surveying the city for viable parking spots, we found certain areas in Delhi which have been very effectively used for parking by the local residents. One that caught our eye was the area beneath the rajouri garden flyover. Though the parking is not properly made and most people don't know that it is an available parking space, the idea was itself unique and a really space efficient one. Now, if our concept "Parking Guidance System" was applied to all such viable under-flyover spaces, we might be able to solve the parking problem of the city while generating huge revenues for the government as well.

ADVANTAGES OF THE PARKING GUIDANCE SYSTEM

A parking guidance system not only saves a lot of time and nerves in finding a free parking space, it also reduces the traffic searching for parking spaces in a car park and therefore reduces the emission of pollutants and noise. The capacity utilization and the acceptance of the car park is increased.

Thus, the Parking Guidance System helps in a variety of ways:

- ## ACCURATE SIZING TO THE REQUIREMENTS

The Parking guidance system has a modular structure. Almost all components of the system are equipped with a basic functionality, which can be extended with additional modules according to customer demand. Whether it is an additional indication of the parking space occupancy in the driving lane or a relay output for controlling a traffic light by one of our controllers. The system can precisely match the given requirements, regardless of whether you want to monitor 10 or 10,000 parking spaces. The chance of an expensive over sizing is practically eliminated.

- ## LOW INSTALLATION COSTS

Time is money. For this reason, great value is placed on a simple and quick installation of the components. For example, we can use spring terminals in the sensor to easily loop the cable through. A cable termination is not required. Furthermore, the ultrasonic sensors can be separated into a connection module and a sensor module. The connection modules are also suitable for harsh environmental conditions and can thus be assembled and electrically connected, while the sensor modules are needed only at the commissioning and can be screwed to the connection module. The cable costs are also pretty low.

- ## EFFICIENT USAGE OF AVAILABLE LAND

Since in modern world, where space has become a very big problem and in the era of miniaturization it has become a very crucial necessity to avoid the wastage of space in modern cities.

- ## SECURITY FROM THEFT AND CAR DAMAGES

An ordered and systematic parking complex with least human interference and 24 hour CCTV surveillance provides security both from theft and any damages being caused to the car.

- ## SPEEDY SYSTEM, SMOOTH FUNCTION

With this guidance system the entire process of vehicle parking becomes hassle-free and speedy. The automated billing system gives the vehicle owner a respite from the constant tension of car parking and makes things really smooth and easy for the parking manager.

- EVERYTHING FROM ONE SOURCE

Single space- or directional sensors, residual space- and open / full / closed displays, visualization and control of the entire car park on a PC, everything from a single source.

- LEAST MANUAL LABOUR REQUIRED

Least possible labour is required for such a smoothly functioning system. A single parking manager would be enough to supervise the entire system.

- PROPER UTILIZATION OF ADVANCED TECHNOLOGY

The system is in accordance to the idea of "Digital India" of PM Mr. Narendra Modi. The project promotes use of advanced and modern technologies for betterment of society and advancement of existing facilities.

- DRIVING AROUND IN SEARCH OF A PARKING SPACE IS ELIMINATED, THEREBY REDUCING ENGINE EMISSIONS.

Reduces air pollution by reducing both the amount of time spent in searching for a parking, and also the distance people have to travel to park their vehicles quite far from their desired locations.

- EASY INTEGRATION OF CUSTOMER SPECIFIC SYSTEMS

The Parking guidance system can be expanded by various input and output modules. The communication protocol is disclosed, providing easy integration of customer specific systems. And if there is no suitable solution available for the problem, one can be developed, starting with individual LED signs, software enhancements in controllers.

PROJECT MANAGEMENT

INTRODUCTION

Project management uses a systematic and disciplined approach to develop software. It consists of all the umbrella activities, which span throughout the software process.

It includes the following activities:

- Estimation
- Project Scheduling
- Risk Management
- Quality Management
- Change Management

Project management involves the planning, monitoring and control of the people, process and events that occur as software evolves from a preliminary concept to an operational implementation. Effective software project management focuses on the four principles: people, product, process and project.

❖ THE PEOPLE:

Software engineering institute has developed a people management capability maturity model (PM-CMM). The people management maturity model defines the key practice areas for software people like: recruiting, selection, performance management, training, compensation, carrier development, organisation and work design and team/culture development.

❖ THE PRODUCT:

Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered and technical and management constraints should be identified. Scope identifies the primary data, functions and behaviours that characterize the product.

❖ THE PROCESS:

A software process provides the framework from which a comprehensive plan for software development can be established.

Framework activities are populated with tasks, milestones, work products and quality assurance points. These activities characterize the software product and the project team.

Umbrella activities i.e. software quality assurance, software configuration management and measurement overlay the process model.

PROCESS MODEL

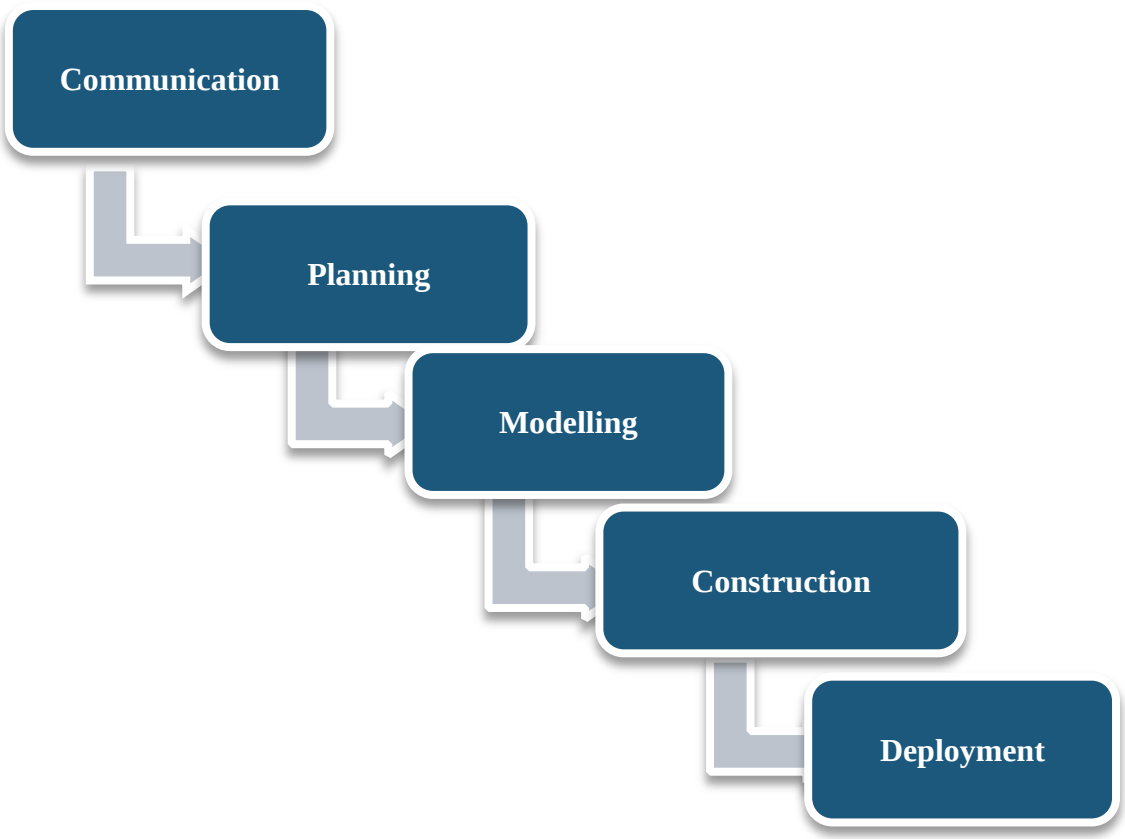
The software model that we are using to develop our project –
"AUTOMATED PARKING SYSTEM" is 'LINEAR SEQUENTIAL MODEL'. This model is sometimes called the CLASSIC LIFE CYCLE or WATERFALL MODEL.
The linear sequential model suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment culminating in on-going support of the completed software.

Since it is not made with respect to a particular organization or person and made according to mindset of different users so requirements of the problem are reasonably well understood i.e. work flows linearly from communication through deployment. As the software is not made for a particular customer so no changes are implemented with respect to the customer, but the requirements are gathered and analyzed by the project team keeping in mind the needs of different customers.

Different phases of the model:-

1. Software requirements analysis:

In this, software engineer understand the nature of a program to be built, he must understand the information domain for the software as well as required function, behaviour, performance and interface. Requirements for both the system and the software are documented and reviewed with the customer.



2. Design:

It has four distinct attributes of a program: data structure, software architecture, interface representation and procedural detail. It is documented and becomes part of the software.

3. Code generation:

Design must be translated into a machine readable form which is done by code generation.

4. Testing:

It focuses on the logical internals of the software, ensuring that all the statements have been tested, and on the functional externals; that is conducting test to uncover errors and ensure that defined input will produce actual results.

5. Support:

This is a phase when software when software will undoubtedly undergo change after it is delivered to the customer. Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment, or because the customer requires functional or performance enhancements. Software support/maintenance reapplies each of the preceding phrases to an existing program rather than a new one.

We are opting for the Linear Sequential Model because:

a. All requirements for the project have been explicitly stated at the beginning. There is very little scope of customer's deviation from current requirements, coding and testing after detailed analysis is much easy.

b. The requirements are not very complex in nature. Hence, the system has low complexity.

c. The probability that the customer will change the requirements is very low. Also a time constraint is provided for software completion, so development team can assume that requirements are almost frozen and move to deployment as soon as possible.

d. The software works in a linear fashion and there is little scope of changing the system once it has been setup.

RISK MANAGEMENT

Risk Analysis and Management are a series of steps that help a software team to understand and manage uncertainty. Many problems can plague a software project. A risk is a potential problem- it might happen or it might not. But regardless of the outcome, it's a really good idea to identify it, assess its probability of occurrence, estimate its impact, and establish a contingency plan should the problem actually occur.

Everyone involved in the software process-managers, software engineers, and stakeholders-participate in risk analysis and management.

Software is a difficult undertaking. Lots of things can go wrong, and frankly many often do. It's for this reason that being prepared- understanding the risks and taking proactive measures to avoid or manage them- is a key element of good software project management.

Recognizing what can go wrong is the first step, called "Risk Identification". Next, each risk is analyzed to determine the likelihood that it will occur and the damage that it will do if it does occur. Once this information is established, risks are ranked, by probability and impact. Finally, a plan is developed to manage those risks with high probability and high impact.

The work product is "Risk Mitigation, Monitoring and Management (RMMM) Plan" or a set of risk information sheets is produced.

RISK STRATEGIES:

1) REACTIVE: A Reactive Strategy monitors the risk project for likely risk and set aside resources to deal with them, should they become actual problems.

2) PROACTIVE: A Proactive strategy begins long before technical work is initiated. Potential risks are identified, their probability impact is assessed, and they are ranked by importance.

TYPES OF RISKS

1) PROJECT RISK: They threaten the project plan. They identify potential budgetary, schedule, personnel, resource, custom potential and requirements problem and their impact on software project. They threaten the project plan.

2) TECHNICAL RISK: They threaten the quality and timeliness of the software to be produced. They identify potential design, implementation, interface verification, and maintenance problem. They threaten the quality and timeliness of software to be produced.

3) BUSINESS RISK: They threaten the viability of the software to be built. They often jeopardize the project or the product and include market risk, strategic risk, management risk and budget risk.

RISK MITIGATION, MONITORING AND MANAGEMENT (RMMM) PLAN

Risk is any event that causes delay in the project. It may occur or may not occur. Risk analysis and management are actions that help a software team to understand and manage uncertainty.

Following steps are being followed by software team to manage risks:

a. Risk is identified.

b. Each risk is then analysed to find its probability of occurrence and the damage that it will do if it does occur.

c. Then each risk is ranked according to probability of occurrence and impact on software.

d. Plans are developed to manage risks with high probability and impact.

Identifying potential risks and developing a plan to manage, monitor and manage risks is of paramount importance. Risk analysis enables to build a risk table by providing detail guidelines in identification and analysis of risk. This is achieved by:

a. Risk avoidance

b. Risk monitoring

c. Risk management and contingency plan.

Sno	Risk	Category	Probability	Impact	RMMM Plan
1.	The team may lose all the project artifacts any time during the project and thus will be unable to deliver the application to the customer. Such an unlikely event may be caused by a hard disk being wiped out by a virus, hard disk failure, etc.	Project Risk	10%	3	To avoid losing the work already completed, the team will have to carry out a necessary backup of database data, source code and documentation. Ensure that backups are made in regular intervals of time.

2.	The duration of project is two months and within this period it is possible that the financial standing of the customer may deteriorate to the extent that the customer would not be able to pay for the project.	Business Risk	10%	2	One possibility is to get as much funding in advance as possible or else find another company involved in the same type of business which may be interested in pursuing the project.

3.	Customer requirements might change. Since our software and system is made in a linear fashion, changing of requirements can be a big problem.	Project Risk	20%	3	SRS should be documented and validated with the customer in advance.

4.	The project may face maintenance and interface problems leading to breakdown of the system due to the malfunction of the gadgets used in the system.	Technical Risk	10% & linearly proportion al to time	2	Timely checking of the functionality of the gadgets and also associating trained staff to manage the gadgets.
5.	The project may have to deal with customers that are illiterate thus lacking them to understand the functionality of the project system and finding it	Business Risk	10%	4	Proper management of staff/attendant could help to reduce this risk. The customer could verbally communicate with the attendant and resolve their difficulty in handling the system.
	difficult to interact with the system.				
6.	Lack of training on tools or insufficient skills for operating the system.	Business risk	30%	3	Staff must be trained to handle working of tools.

7.	Team dissension/ Lack cohesion	Project Risk	10%	3	Some guidelines and rules should be set describing how to deal with each other.

IMPACT:-

1 - Catastrophic

2 - Critical

3 - Marginal

4 - Negligible

SOFTWARE PROJECT PLAN

Software project scheduling is an activity that distributes estimated effort across the planned project by allocating the effort to a specific software engineering tasks.

When you develop a schedule, compartmentalize the work, represent the task interdependencies, allocate effort and time to each task, define responsibilities for the work to be done, and define outcomes and milestones.

In order to build a complex system, many software engineering tasks occur in parallel and result of work performed during one task may have a profound effect on work to be conducted in another task. These interdependencies are very difficult to understand without a schedule. Its also virtually impossible to progress on a moderate or large software project without a detailed schedule.

1. Timeline Chart:

When creating a software project schedule, the planner begins with a set of tasks. If automated tools are used, the work breakdown is input as a task network or task outline. Effort, duration and start date are then input for each task outline. In addition, tasks may be assigned to specific individuals.

As a consequence of this input a Timeline Chart also called a Gantt chart is generated. It enables you to determine what tasks will be conducted at a given point of time. Following is the timeline chart for our project:

S.NO.	PROCESS/PHASE	START DATE	END DATE
1.	Requirements gathering	1.2.17	8.2.17
2.	Requirement analysis	9.2.17	12.2.17
3.	Risk management plan	13.2.17	19.2.17
4.	Data dictionary preparation	25.2.17	1.3.17
5.	FPA calculation	16.3.17	20.3.17
6.	DFD preparation	20.2.17	24.2.17
7.	Database design	2.3.17	9.3.17
8.	Screen design	10.3.17	15.3.17
9.	Pseudo code	21.3.17	28.3.17
10.	** Project Management	1.2.17	28.3.17

** Project Management is an Umbrella Activity, which is carried throughout the duration of the project.

FUNCTIONAL POINT ANALYSIS

FPA is a standard metric for the relative size and complexity of a software system, originally developed by Alan Albrecht of IBM in the late 1970s.

Function points (FPs) can be used to estimate the relative size and complexity of software in the early stages of development analysis and design.

The size is determined by identifying the components of the system as seen by the end-user: the inputs, outputs, inquiries, interfaces to other systems, and logical internal files.

The components are classified as simple, average, or complex.
All of these values are then scored and the total is expressed in Unadjusted FPs (UFPs). Complexity factors described by 14 general systems characteristics, such as reusability, performance, and complexity of processing can be used to weight the UFP. Factors are also weighed on a scale of 0 - not present, 1 - minor influence, to 5 - strong influence. The result of these computations is a number that correlates to system size.

Although the FP metric doesn't correspond to any actual physical attribute of a software system (such as lines of code or the number of subroutines)

It is useful as a relative measure for comparing projects, measuring productivity, and estimating the amount development effort and time needed for a project.

Complexity Adjustment Values Table

S.No.	Questions	VAFs
1	Does the system require reliable backup and recovery?	3
2	Are specialized data communications required top transfer information to or from the application?	3
3	Are there distributed processing functions?	3
4	Is performance critical?	4
5	Will the system run in an existing, heavily utilized operational environment?	3
6	Does the system require online data entry?	2
7	Does the online data enquiry require the input transaction to be built over multiple screens or operations?	3

8	Are the ILFs updated online?	2
9	Are the inputs, outputs, files or inquiries complex?	3
10	Is the internal processing complex?	4
11	Is the code designed to be reusable?	3
12	Are conversion and installation included in the design?	3
13	Is the system designed for multiple installations in different organizations?	4
14	Is the application designed to facilitate change and ease of use by the user?	3
	∑F(i)	43

FUNCTION POINT EVALUATION:

ILF Complexity Matrix

S.No	Data File	No. of Data Fields	No. of Records	Complexity
1.	Vehicle database	5	2	Low
2.	Sensor file	1	1	Low

Input Screen Complexity

S.No	Input Screen	No. of fields	No. of Files	Complexity
1.	Payment	5	1	Low
2.	Camera feed	3	1	Low

Output Screen Complexity

S.No	Output Screen	No. of Data Fields	No. of Records	Complexity
1.	Bill	5	1	Low
2.	Token	3	1	Low

Inquiry Screen Complexity

S.No	Query Screen	Complexity
1.	Vehicle Info	Low
2.	Parking lot status	Low

EIF Complexity Matrix

S.No	Data File	Complexity
1.	Records	Low

Unadjusted Function Point Table

S.No	FP Fundamental	Low	Average	Complex	Total
1.	No. of Inputs	2*3	0*4	0*6	6
2.	No. of Outputs	2*4	0*5	0*7	8
3.	No. of Inquiries	2*3	0*4	0*6	6
4.	No. of ILFs	2*7	0*10	0*15	14
5.	No. of EIFs	1*5	0*7	0*10	5
					39

Function Point Calculation

FP = UFP*(0.65+0.01*∑F(i))

= 39*(0.65+0.01*43)

= 39*1.08

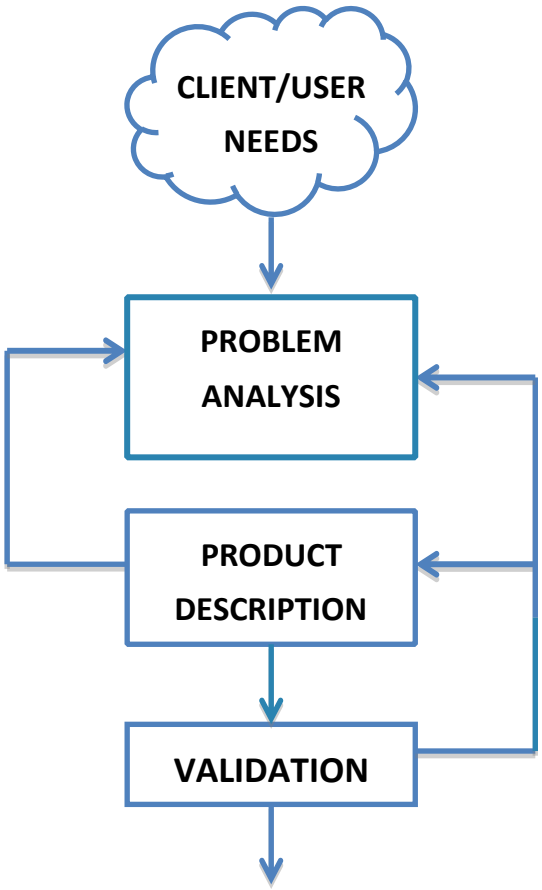
= 42.12

REQUIREMENTS GATHERING AND ANALYSIS

REQUIREMENT PROCESS INTRODUCTION

Requirements Process is the sequence of activities that need to be performed in the requirements phase and it culminates in producing a high quality document containing the software requirements specification (SRS). The requirement process consists of three basic tasks:

1. Problem or Requirements Analysis
2. Requirements Specifications
3. Requirements Validation



Problem Analysis starts with a high-level problem statement. During analysis the problem domain and the environment are modeled in an effort to understand the system behaviour, constraints on the system, its inputs and outputs etc. The basic purpose of this activity is to obtain a thorough understanding of what the software needs to provide. The understanding obtained by problem analysis forms the basis for requirements specification.

Requirements Specification focuses on clearly specifying the requirements in a document. Issues such as representation, specification languages and tools are addressed during this activity. As analysis produces large amounts of information and knowledge with possible redundancies, properly organizing and describing the requirements is an important goal of this activity.

Requirements Validation focuses on ensuring that what has been specified in the SRS are indeed all the requirements of the software and making sure that the SRS is of good quality. The requirements process terminates with the production of the validated SRS.

To communicate with the client and the end-users in a systematic and effective manner, a questionnaire was prepared.

QUESTIONNARE

The Aim of the Questionnaire is to collect information on the methods and practices for software development. Your Input is extremely important to identify gaps and desired data that will help in driving future tools and methods that may better support your activity. **The information collected is completely confidential. We shall never share or release your private information to an unaffiliated third party.**

Ques1:- Do you easily finding parking space near your home?

A. Yes

B. No

Ques 2:- Which mode of parking do you prefer?

A. automatic

B. semi-automatic

C. manual

Ques 3:- Do you want to know the vacant space available for parking and Why?

_____

_____

Ques 4:- How do you register for your entry in the parking?

A. token
B. slip
C. card

Ques 5:- Would you be willing to incorporate better and smarter technologies for improving the existing parking system?

_____

Ques 6:- What is the average time taken to park the vehicle into the parking space?

_____

Ques 7:- How do you think the charges of parking should be?

A. Hour based
B. Entry based

Ques 8:- Are you willing to pay extra money if you could avail parking near your homes or desired locations? _____

Ques 9:- Do you know any land space around your house that can effectively be turned into a parking lot?

_____

Ques 10:- The preferable mode of payment:

A. Cash B. Credit Card

C. Debit card D. Mobile Banking

Ques 11:- Extra Benefits provided (if any)?

_____

Ques 12:- Complaints (if any).

_____ Ques 13:- Suggestions for Improvement.

_____

A SRS is a requirement specification for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that descried all the interactions that the user will have with the software.

The following subsections of the Software Requirement Specification provides an over view of the Initial SRS.

- ## PURPOSE:

  The **parking guidance system** is a mechanical system designed to minimize the area required for parking vehicles. The project also minimizes the manual efforts needed to manage to system. It ensures security to the vehicles.

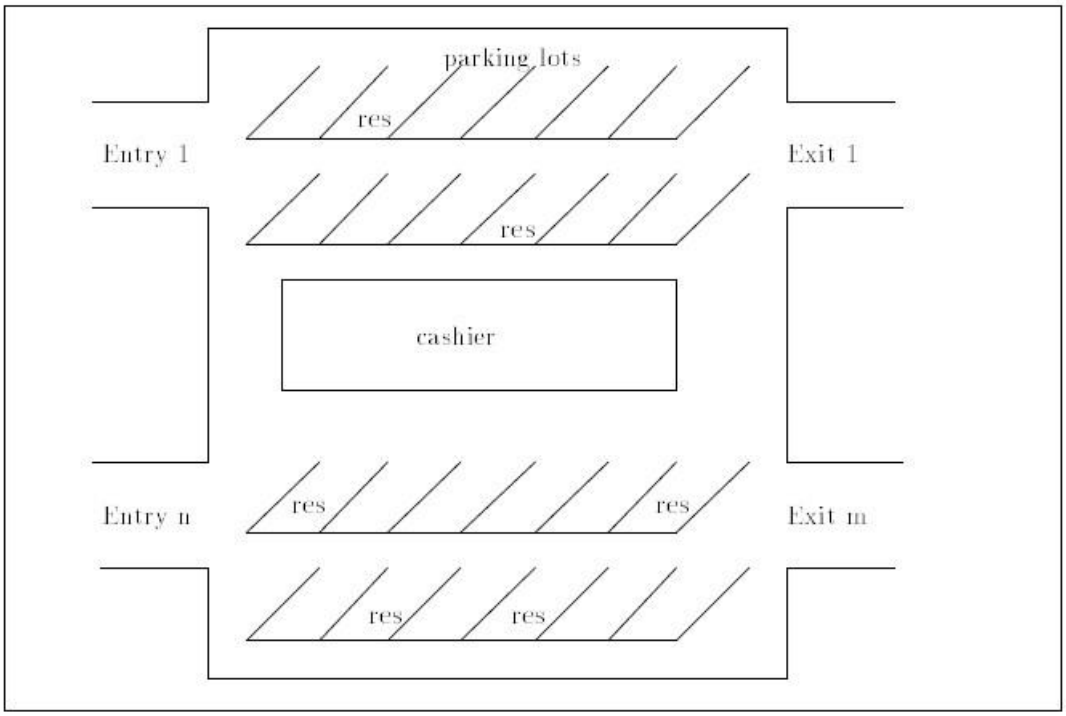
- ## FUNCTIONALITY:

1. The project must provide the best parking space to the customer according to his/her requirements by checking all the available space and choosing the optimum space.
2. It should assure proper safety to the vehicles.
3. The system should store appropriate details of the vehicles entered in the parking lot.
4. The system should be user friendly and easy to use.
5. The project should provide accurate navigation hoardings to reach the allocated parking space.
6. A LCD display to show number of available parking space and vacant space.
7. A slip generator that issue entry tickets.
8. A sensor system that tells the system manager about the vacant space and user get the position (block and row no.) on entry ticket.
9. A camera that took the photos of vehicles to keep track and store it in database.
10. Automatic boom barrier that opens up after issue of the entry ticket.
11. A billing management that generates a bill on the exit gate and open the boom barriers after payment on the machine.

After further analysis, requirement gathering and validation from the client the final SRS was defines as:

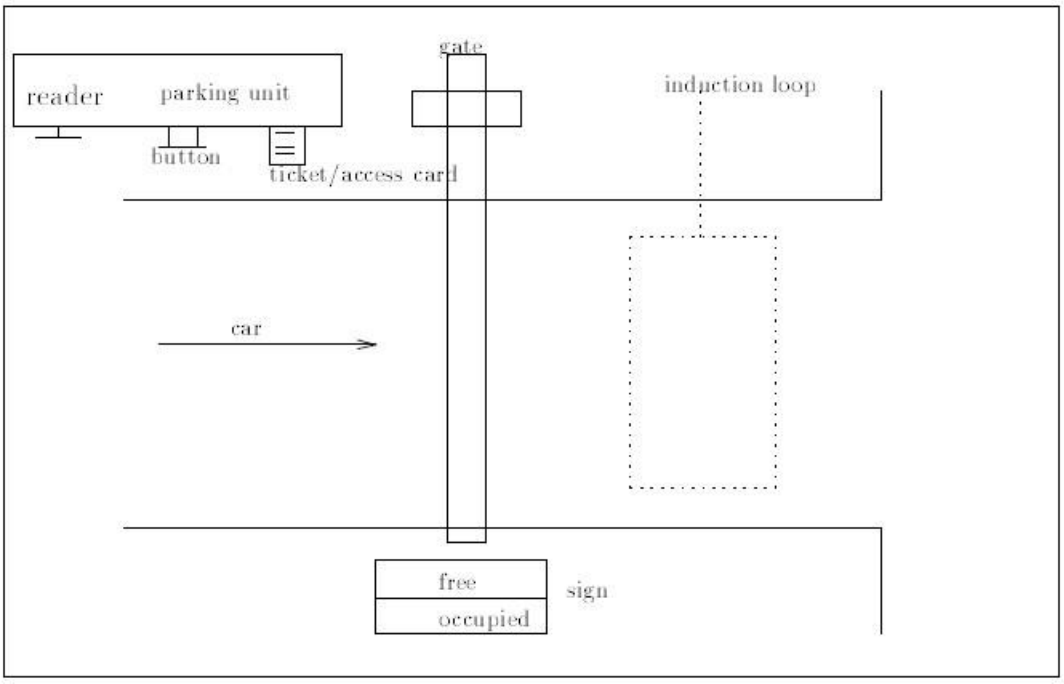
DEFINITIONS

Parking garage consists of n entries and m exits. There are k parking spaces. The maximal number of parking spaces is 1000.



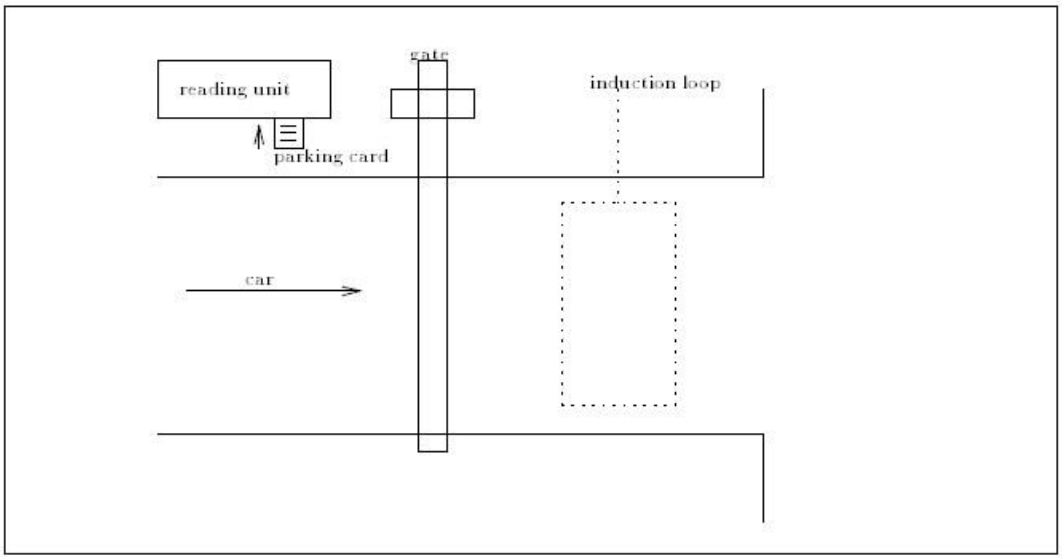
Parking Garage

An entrance consists of a gate, a state display showing whether any parking space is available, a ticket machine with a card reader, and an induction loop. The ticket machine consists of a request button, a unit for the output of the tickets and a card reader.



Entrance

The exit consists of a gate, a ticket reader, and an induction loop that is behind the gate.



Exit

OVERALL DESCRIPTION

To give a short overview of the functionality of the "Parking Guidance System", the following user scenarios are provided:

Entry:

1. A driver pushes the button at the ticket machine. If the parking garage is full, nothing happens (LCD displays zero vacancy).
2. The ticket machine writes the time and the spot that has been fixed for the vehicle on the ticket and delivers the ticket to the driver.
3. The surveillance camera takes a photo of vehicle's number plate to store in the database. The gate will open when the driver takes the ticket.
4. The parking status on the LCD is changed outside where the entry for no. of vehicles is increased.
5. The driver enters the parking garage.
6. After the car passes the loop, the gate is closed.
7. The driver parks the vehicle and leaves the parking garage. The sensor above the newly parked vehicle turns from green to red.

Exit:

1. The driver returns to his car and drives to an exit station. The sensor above the vehicle turns to green once the vehicle has left.
2. The driver puts the ticket in the ticket reader.
3. The ticket reader checks the ticket and determines what amount of money has to be paid by the driver (hourly basis) and will display it on the screen.
4. Once the amount has been paid, the gate will open.
5. After the car passes the loop, the gate is closed.
6. The parking status on the LCD is changed outside where the entry for no. of vehicles is decreased.

PRODUCT PERSPECTIVE

The software system is an embedded system. The software system should control:- Each Entrance:

- a ticket machine
- a gate
- a card reader
- induction loop
- state display Each Exit:
- a ticket reader a gate
- induction loop
- cash register

Control Unit:

- All administration controls.
- Camera Surveillance.

- Database and Sales reports access SCOPE:
- To make this system more efficient and user friendly.
- To lower the parking fare
- To make the system more applicable to the Indian Scenario 1) User Interface:

The User of this software system will be the parking manager or supervisor. He will be provided with the Graphical User Interface, there is no command line interface for any functions of the software. The user will be able to access surveillance tapes, sales reports, and vehicle database and will also have some privilege commands like vehicle plate recognition, manual operation (emergency situations) etc.

2) Hardware Interface:

For running the software, the requirements are as follows:

- Processor – Pentium 4 or above
- RAM – 512Mb or above
- Hard Disk Drive – 40Gb or above
- Screen Resolution- at least 800*600 for proper or complete viewing of screen

Graphic Card – GeForce4 Ti 128Mb 4200

Setup requirements:

- Visual Control Centre(PC or Mainframe depending upon requirement)
- Zone Controller ZK300 (sensor control)
- Data Concentrator DC300 (vehicle counting)
- Single Space Sensors(Ultra Sonic)
- External Lamp
- LCD Displays
- Ticket Vending Machine
- CCTV Cameras
- Storage Disks (Surveillance Tapes)

3) Software Interface

Software required to make working of product is:

- Operating System-Windows XP or above
- Atmel Studio (any Embedded C compiler), MySql, Microsoft Visual Studio, TTL application (for wireless communication between various controllers)

4) Communication Interface:

The communication between the different parts of the software system as they depend on each other. However, in what way the communication is achieved is not important for the system and is therefore handled by the underlying operating system.

Simple telephone cables or XBEE or Bluetooth or USART/UART can be used for communication among various controllers or between controllers and sensors.

5) Memory Constraints:

At least 256Mb RAM and 2GB space on hard disk will be required for running the software. The surveillance tapes will require additional storage space.

6) Site Adaptation Requirements:

- Setting up of new hardware and cable lines with least modification to the existing structure.

- Separate and preferably central space for main controller.
- Incorporating the already established database.

PRODUCT FUNCTION

The software system should control the state display, gates, ticket machines and ticket readers.

- If the request button is pressed the driver should get a ticket and the gate should be opened if there is a parking space available.
- Entering a ticket in a ticket reader should open the exit gate once the fee was paid.
- The gates should be closed after the car has passed the induction loop.
- The state display should show the actual status of occupancy.

  For testing and maintenance of the system, there is the possibility of entering a certain state of occupation with this privilege only available for the administrator.

The parking supervisor is not part of the software system.

CONSTRAINTS

- There is an established backup for the system.
- GUI features (Visual C++) etc are available
- The system is single user system (The system will be run by a single parking manager or supervisor)

USER CHARACTERISTICS

The system users (drivers) should not require special training.

ASSUMPTIONS AND DEPENDENCIES

- One assumption about the product is that it will always be used on a system that has enough performance. If the system does not have enough hardware resources available for the application, there might be scenarios where the application does not work as intended.

- The system must be able to respond to the database within reasonable time.

Assumptions about the parking garage:

- Every parking space can be reached from any entrance.
- Every exit can be reached from each parking space.
- No entrances are convertible to exits and vice versa.
- Emergency situations (e.g. fire) will not be considered here.
- The number-plate recognition system works perfectly.
- The user or the driver works in a systematic manner as directed.

SPECIFIC REQUIREMENTS

User Class and Characteristics:

There is only one type of user for this system, the Administrator or the parking supervisor. The Administrator keeps track on all the movements of system like entry, exit, payment etc.

He has access to specially designed sales reports and to vehicle database and parking layout.

He also has privileged instructions like camera footage, manual operation, number plate recognition.

Functional Requirements:

Functional requirements define the capabilities and function that a system must be able to perform successfully. The functional requirements of this parking guidance system include:

- The system should control the entries and exits of a parking garage.

  The system has to guarantee that no more than k(max) cars are in the parking garage. The default value for k is 10000.
- The system should support ‘n’ entries and ‘m’ exits. The system has to handle simultaneous entries and exits.
- The total number of parking spaces can be written with the control unit. With the control unit it is possible to enter the total number of parking spaces currently allocated.
- The state display should represent the state of occupancy of the public parking spaces. It should display 'free' if there is a parking space available at that moment. It should display 'occupied', if there is no parking space available at that moment.
- Every driver should get a ticket at the entrance only if there is a parking space available.

- Driver presses the button once.
- The gate will open if the ticket is handed out.
- If more than one car wants to enter the parking garage through different entry stations, the system has to synchronize all stations.
- At the exit a car arrives and driver puts a ticket into the ticket reader. If the ticket is money is demanded.
- If the induction loop is crossed, the gate should close.
- If several cars leave the parking garage at the same time the PGCS has to synchronize all actions.

External Interface Requirements:

- Apart from the control unit there is no need for a user interface.
- There has to be hardware interfaces to the ticket machines, the ticket readers, the gates, the sensors and the loops. The system will get signals from and will send signals to these devices.

Performance Requirements:

- After a car has passed the induction loop the gate has to close within 5 sec.
- If a driver requests a ticket and there are free parking spaces available, he will get the ticket within 3 sec.
- If a gate opens, it will remain open at least 5 sec.
- Only one car should pass through the gate each time it opens.
- All changes to state variables at entry or exit station should happen within 5 sec.
- For each car that enters the parking garage there is a parking space available.
- Responses to queries should not take more than 7 seconds to load on the screen.
- The camera footage should be displayed in real time with no lag or jitter.

System Attributes:

Availability: The system has to be available 24 hours/day. The parking garage won't be closed at any time.

Security: No tickets other than the tickets of this parking garage should be accepted by the ticket reader.

Maintainability: It should be easy to integrate the cashier into the software system.

Reliability: The reliability that the system gives the right results on the given input.

Transferability/Conversions: Not Applicable

Caution: Not Applicable

DATA FLOW DIAGRAM

DEFINITION:

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

COMPONENTS OF DATA FLOW DIAGRAM:

External Entity:

An External Entity is a source or destination of a data flow which is outside the area of study. Only those entities which originate or receive data are represented on a business process diagram. They may represent an organization, customers or other organization.

Data Stores:

Data Stores represent stores of data within the system, for example, computer files or databases. Data stores may be long-term files such as sales ledgers, or may be short-term accumulations: for example batches of documents that are waiting to be processed.

Processes:

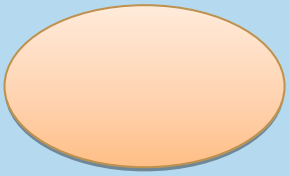
Processes represent activities in which data is manipulated by being stored or retrieved or transferred in some way. A process shows a transformation or manipulation of data flows within the system. In other words, we can say that process transforms the input data into output data. Circles stand for a process that converts data into information.

Data Flows:

A Data Flow shows the flow of information from its source to its destination. A data flow is represented by a line, with arrowheads showing the direction of flow. Information always flows to or from a process and may be written, verbal or electronic. Flow of data in the system can take place:

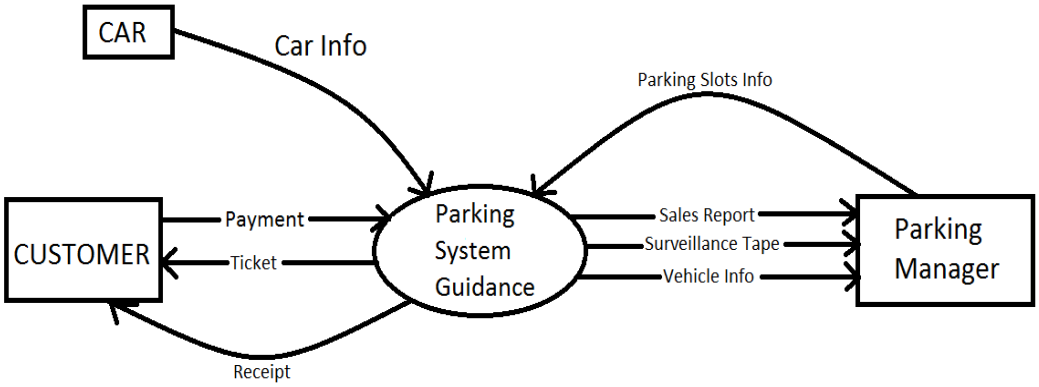
- Between two processes
- From Data Store to a Process
- From a Process to a Data Store
- From a Source to a process end

SYMBOLS USED IN DFD:

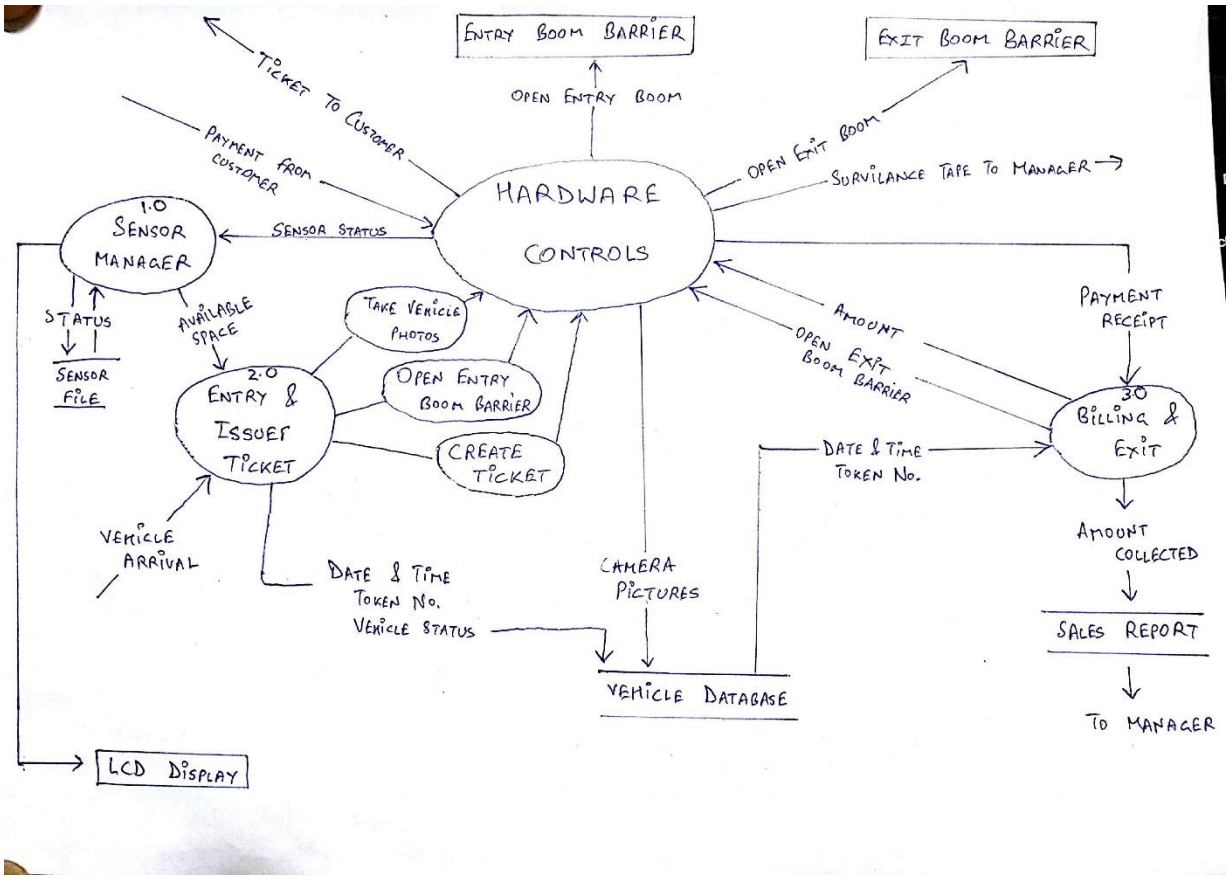
S.No.	Symbol	Name	Function
1.		Circle/ Oval	Represents Process or transformation function
2.		Arrow	Represents

			Input/output Data Flow.
3.		Rectangle	Represents External Entity i.e. source or sink
4.		Line	Represents External Database File

LEVEL 0 DFD/ CONTEXT DIAGRAM

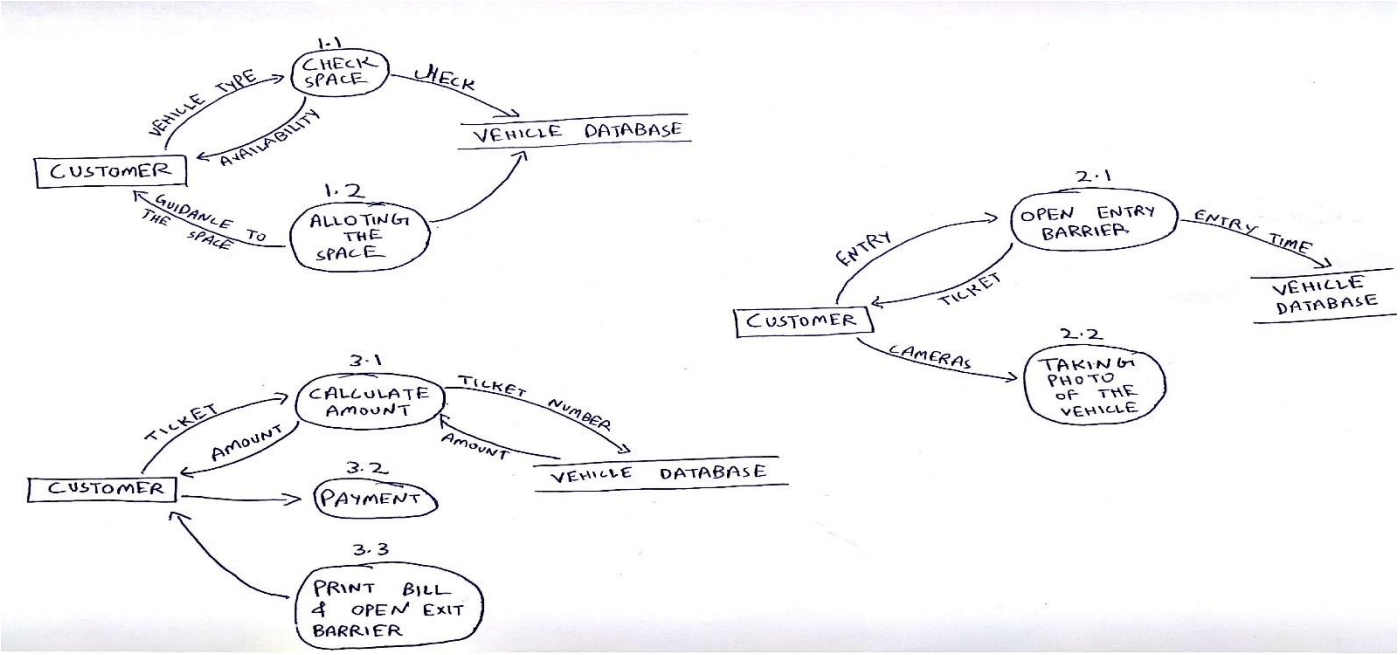


LEVEL 1 DFD



LEVEL 2 DFD

(HARDWARE CONTROL)



DATA DICTIONARY

The data dictionary provides an organised approach for representing the characteristics of each data object and control item. It has been proposed for describing the content of objects defined during structured analysis.

A Data Dictionary is very important in the software development process because of the following reasons:

1) A Data Dictionary lists standard terminology for use by an engineer working on a project.

2) The Dictionary provides the analyst with means to determine the definition of different data structures in terms of their component elements.

The format of Data Dictionary includes the following information:-

- Name-the primary name of the data or control item, the data store or an external entity.
- Alias-other names used for first entity.
- Description-a notion for representing content.
- Type-type of the data.

DATA ITEM	DATA TYPE	DATA LENGTH	DATA DESCRIPTION
CUSTOMER			
Customer Name	String	20	Contains the customer's name
Customer Id	Alphanumeric	23	Contains customer's email-id
Vehicle number	Alphanumeric	15	Contains the customer's vehicle number
Vehicle type	Alphabetic	20	Contains the type of vehicle parked
Phone Number	Numeric	10	Contains the customer's phone number
PARKING LOT			
Block	Alphanumeric	5	Contains the address of the available space in the parking lot.
Boom barriers status	Boolean	1	Tells whether the barrier is up or down
Sensor status	Boolean	1	Tells whether a vehicle is parked at a current spot or not.
TICKET			
Ticket_no	Alphanumeric	10	Contains the ticket number
Date	Date/time	15	Contains the date

Entry Time	Date/time	15	Contains the time a vehicle enters the parking lot

Exit Time	Date/time	15	Contains the time the vehicle exits from the parking lot.
Duration	Date/time	15	Contains the time for which the vehicle was parked
Per hour charges	Numeric	10	Contains the per hour rate of occupying a parking spot
Amount	Numeric	10	Contains the total amount the customer has to pay
LCD DISPLAY			
LCD _total	Numeric	4	Displays the total capacity of the parking lot
LCD_available	Numeric	4	Displays the available number of parking spots in the lot.
LCD_occupied	Numeric	4	Displays the occupied spots in the parking lot.
VEHICLE DATABASE			
Vehicle_no	Alphanumeric	10	Contains the registration number of the vehicle

Vehicle_type	Alphabetical	20	Contains the type of vehicle to be parked
Vehicle condition at entry	Picture format		Contains the images of the vehicle at the entry check point
Vehicle condition at exit	Picture format		Contains the images of the vehicle at the
			exit check point.
Duration	Numerical	10	Contains the time for which the vehicle was parked.

DESIGN

INTRODUCTION

The design activity begins when the requirements document for the software to be developed is available and the architecture has been designed. During design we further refine the architecture.

Software design is a process of problem solving and planning for a software solution. After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for a solution. It includes low-level component and algorithm implementation issues as well as the architectural view. The design of a system is a blueprint or a plan for a solution for the system. Here we consider a system to be a set of modules with clearly defined behaviour, which interact with each other in a defined manner to produce some behaviour or services for its environment. A module of a system can be considered a system, with its own modules. The design of a system is correct if a system built precisely according to the design satisfies the requirements of that system.

A design should clearly be verifiable, complete (implements all the specifications), and traceable (all design elements can be traced to some requirements). However, the two most important properties that concern designers are efficiency and simplicity. Efficiency of any system is concerned with proper use of scarce resources by the system. Simplicity is perhaps the most important quality criteria for software systems.

Maintenance of the software is quite expensive. The more simple the software, the more easily it can be maintained.

The design activity mainly focuses on the following major areas of concern:

COMPONENT LEVEL DESIGN: It establishes the algorithmic detail required manipulating the data structures, effect communication between software components via their interfaces, and implement the processing algorithms allocated to each component i.e. it transforms structural elements of software architecture into a procedural description of software components.

INTERFACE DESIGN: It deals with the process of developing a method for two or more modules in a system to connect and communicate. It describes how the software communicates with itself, with systems that interoperate with it, and with the users who use it.

ARCHITECTURAL DESIGN: It defines the relationship among the major structural elements. Here the main objective is to develop a modular structure and represent the control relationship between the modules

DATA DESIGN: It is the first and most important Design activity. It transforms the information domain model created during analysis into the data structures that will be required to implement the software. Hence, Data Design focuses on the definition of data structures.

DATA DESIGN

VEHICLE:

S.No	**Field name**	**Data Type**	**Length**	**Description**	**Mandatory/ Optional**	**Key**
1.	Vehicle no.	Alphanumeric	10	Contains the registration no. of the vehicle	Mandatory	
2.	Vehicle type	Alphabetical	20	Contains the type of vehicle to be parked	Mandatory	-
3.	Ticket no.	Alphanumeric	10	Contains the ticket no.	Mandatory	Primary key
4.	Vehicle condition at entry	Picture format		Contain the image of the vehicle at the entry check point	Optional	-
5.	Vehicle condition at exit	Picture format		Contain the image of the vehicle at the exit check point	Optional	-
6.	Time of entry	Date/time	15	Contains the time the vehicle enters	Mandatory	-
7.	Time of entry	Date/time	15	Contains the time the vehicle exits	Mandatory	-

SENSOR:

S.No.	Field name	Data Type	Length	Description	Mandatory/ Optional	Key
1.	Sensor id	Numeric	10	Contains the id of the sensor	Mandatory	Primary Key
2.	Sensor on/off	Boolean	1	Checks if the sensor is on/off	Mandatory	-

SALES REPORT:

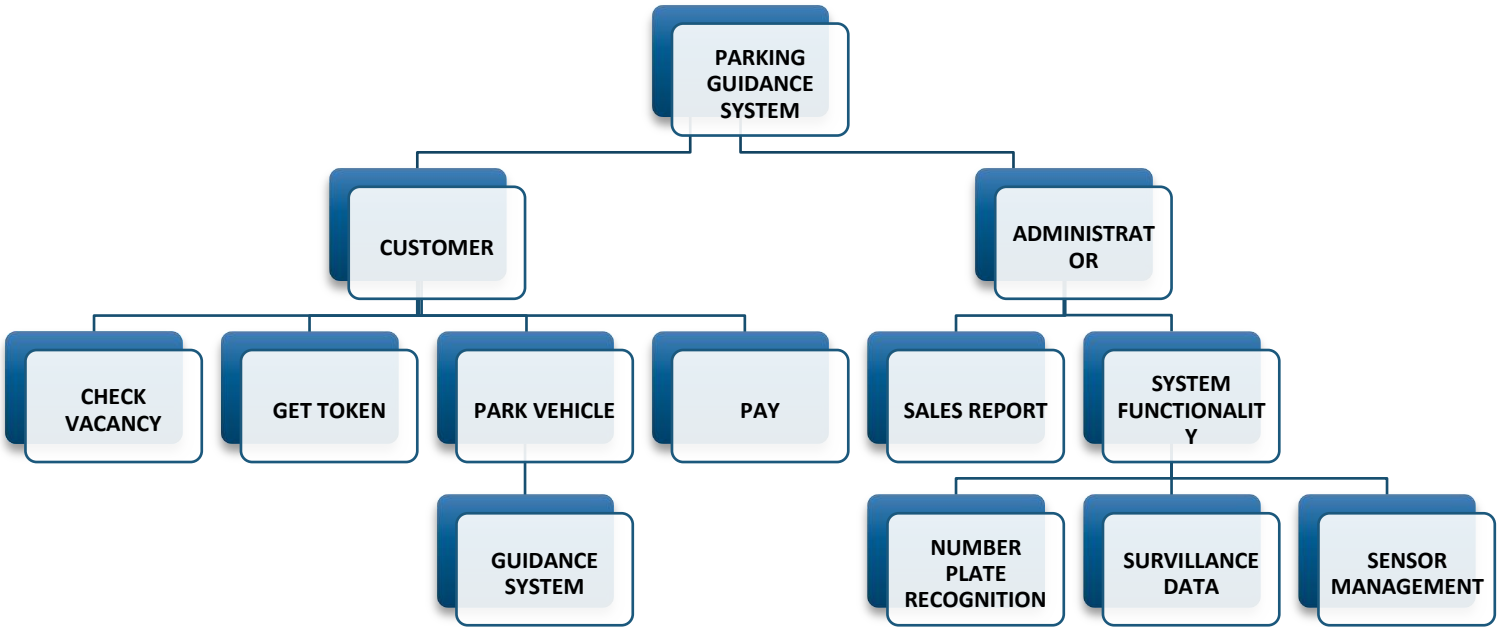
S.No	Field name	Data Type	Length	Description	Mandatory/ Optional	Key
1.	Date	Date	10	Contains the date	Mandatory	-
2.	Ticket no.	Alphanumeric	10	Contains the ticket no.	Mandatory	Foreign Key
3.	Duration	Date/Time	15	Displays the duration for which the vehicle was parked.	Mandatory	-
4.	Estimated amount	Numeric	10	Amount calculated as per the duration	Mandatory	-
5.	Per hour cost	Numeric	10	Cost as per the vehicle	Mandatory	-

SURVEILLANCE TAPES:

S.No	Field name	Data Type	Length	Description	Mandatory/ Optional	Key
1.	Date	Date	10	Contains the date	Mandatory	-
2.	Camera no.	Alphanumeric	10	Contains the camera no.	Mandatory	Primary Key

3.	Time	Date/Time	15	Displays the time.	Mandatory	-

ARCHITECTURE DESIGN



INTERFACE DESIGN

PARKING SYSTEM

SOFTWARE SYSTEM

HOME SCREEN:

SURVEILLANCE TAPES:

VEHICLE DATABASE/PARKING LAYOUT:

Vehicle Number	Parking spot	Entry Time
DL01J-1125	1	01:34:00
DL01K-1344	2	01:36:00
DL01J-1000	3	02:30:00
HR02K-1770	4	05:50:00

SALES REPORT:

Vehicle Number	Entry Time	Exit Time	Duration	Rate per hour(Rs)	Amount(Rs)
DL01J-1125	01:34	03:22	01:48	30	60.00
DL01K-1344	01:36	04:28	02:54	30	90.00
DL32AA-1234	02:15	06:10	03:55	30	120.00
HR53D-1432	02:19	08:09	05:50	30	180.00

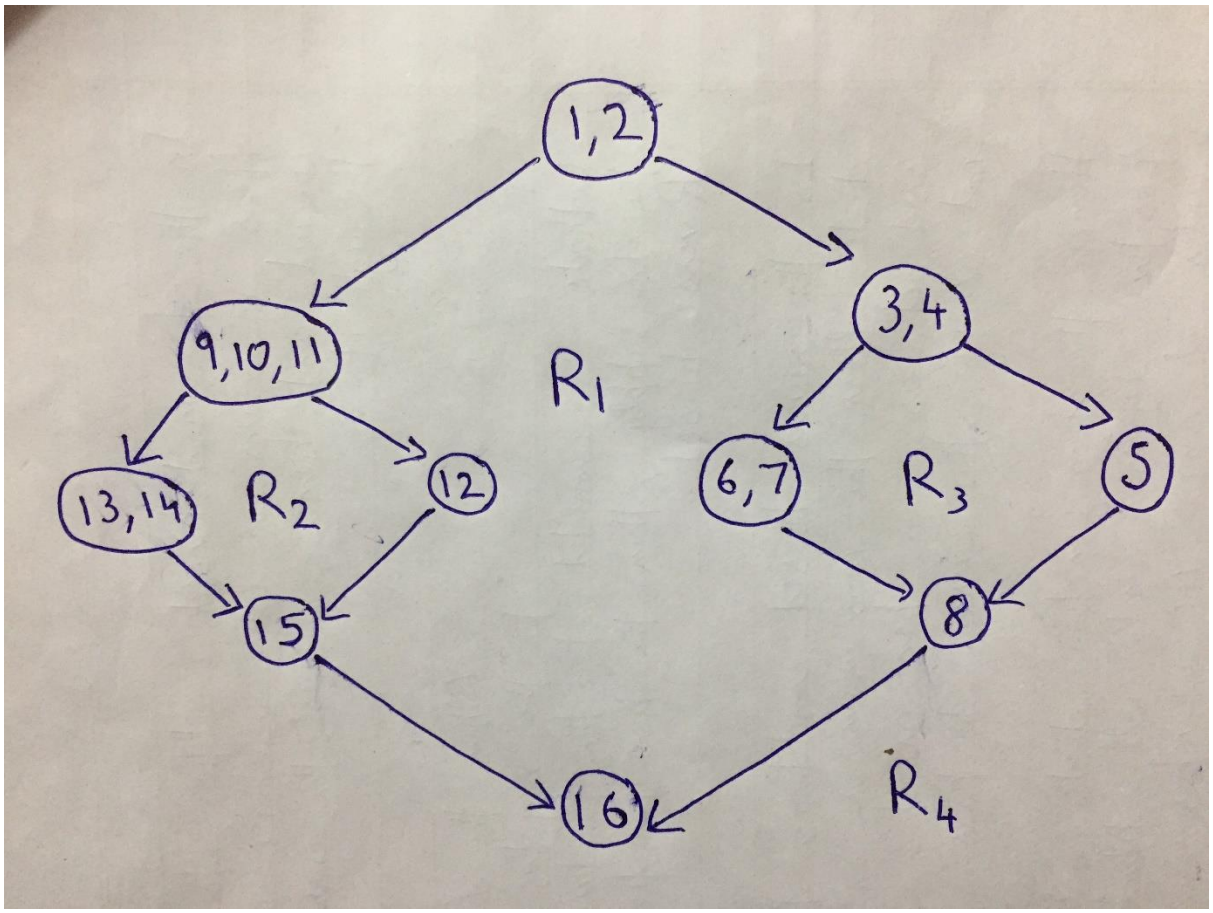
DL21QQ-3425	05:43	07:30	01:47	30	60.00
MP12L-1236	07:33	08:11	00:38	30	30.00
JH11H-3636	07:39	14:20	06:41	30	210.00
PB12S-4567	09:08	10:00	00:52	30	30.00
DL98K-9876	09:10	12:02	02:52	30	90.00
HP12K-1345	12:34	20:28	07:54	30	240.00

Testing

CYCLOMATIC COMPLEXITY

```
void Availability(int num_of_wheels)
1.  {
2.  if(num_of_wheels==4)
3.  {
4.  if(cars_in_park<50)
5.    cout<<“Parking available”;
6.  else
7.    cout<<“Parking unavailable”;
8.  }
9.  else
10. {
11. if(others_in_park<40)
```

```
12.     cout<<“Parking available”;
13.   else
14.     cout<<“Parking unavailable”;
15.   }
16.   }
```



Cyclomatic Complexity = Number of edges - Number of nodes + 2

Therefore, Cyclomatic Complexity = 12 - 10 + 2 = 4

Independent Paths are -

1. 1,2,3,4,5,8,16
2. 1,2,3,4,6,7,8,16
3. 1,2,9,10,11,12,15,16
4. 1,2,9,10,11,13,14,15,16