GET YOUR TICKET

SOFTWARE ENGINEERING PROJECT REPORT

(Submitted in Partial Fulfillment)

As a part of the curriculum of

B.SC (H) COMPUTER SCIENCE

From



Mata Sundri College for Women

University of Delhi

SUBMITTED BY:

Sukhleen kaur – 17044570037

Vanika kamboj – 17044570038

B.SC (H) COMPUTER SCIENCE

IV SEMESTER

Mata Sundri College for Women

Mata Sundri lane , New Delhi - 110002

CERTIFICATE

This is to certify that the project entitled, “GET YOUR TICKET”, an online railway reservation system submitted by Vanika kamboj and Sukhleen Kaur of Mata Sundri College for Women in partial fulfillment of the requirements of Bsc (H) Computer Science, IV semester.

It has been completed under the supervision of Ms. Ashema Hasti from Mata Sundri College for Women, Delhi University.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ms. Ashema Hasti

(Supervisor)

ACKNOWLEDGEMENT

We would like to express my special thanks of gratitude to my teacher Ms. Ashema Hasti, who has provided us the opportunity and motivation to gain knowledge throughout the course of the project . We are also thankful to Mr. Suresh for providing us the help of computer laboratory, which is a valuable input resource for this project. We are very thankful to our teacher for helping us and guiding us in every step. We are thankful to all our friends and other people who have directly or indirectly helped us during the preparation of this project.

Thank you.

Sukhleen Kaur (17044570037)

Vanika Kamboj (17044570038)

ABSTRACT

The railway reservation system is software for the purpose of reserving train seats at any time and from anywhere. This website provides us complete information about a train between specified source and destination. This website gives current status of reservations of particular train, fares for different classes of train and also waiting status. For this website, a visitor must register to avail the service. In this system, train records are maintained and retrieved. Management team has complete access to database and can add train or cancel train or add station for a particular train or skip a station for a train. There are several payment options for user like credit card, debit card and a user can also cancel ticket. This website also provide familiar and attractive interface.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **SNO.** | **CONTENTS** | **PAGE NO.** |
| 1. | Problem Statement | 6 |
| 2. | Software Lifecycle Model | 7 |
| 3. | Project Scheduling | 62 |
| 4. | Timeline Chart | 64 |
| 5. | Software Requirements Specification | 9 |
| 6. | Context Level Diagram | 28 |
| 7. | DFD Level 1 | 29 |
| 8. | DFD Level 2 | 30 |
| 9. | Data Dictionary | 38 |
| 10. | Software metrics | 66 |
| 11. | Effort Estimation Using COCOMO Model | 68 |
| 12. | Risk Analysis | 72 |
| 13. | ER Diagram | 52 |
| 14. | Data Design | 53 |
| 15. | Component level Design | 59 |
| 16. | Use Case Diagram | 41 |
| 17. | Use Case Description | 42 |
| 18. | Testing | 78 |
| 19. | References | 12 |
| 20. | Annexures | 83 |

PROBLEM STATEMENT

GET YOUR TICKET is a software system that provides its users information about the route , booking & cancellation of ticket , departure time, arrival time , number of trains available and other such information. Store and retrieve information about the various transactions related to rail travel is also available. The system will keep track of all its passengers and thus schedule their journey accordingly. It maintains records of passengers travelling in the different trains on different dates reaching different destinations in the system. User can enquire about the PNR status, seats available and trains on a route. It uses user friendly interface to management team and customers. User can check the timings of different trains and can book a seat in any train by paying online, can cancel tickets if needed and can check booking history.

PROCESS MODEL

“Get your ticket” follows incremental process model .

The incremental model combines elements of linear and parallel process flows. The incremental model applies linear sequences. Each linear sequence produces deliverable increments of the software in a manner that is similar to the increments produced by an evolutionary process flow.

Test

Code

Design

Analysis

Inc 1 delivery of Inc 1

Designn

Test

Code

Analysis

Inc 2 delivery of Inc 2

Test

Code

Design

Analysis

Inc 3 delivery of Inc 3

Test

Code

Design

Analysis

Inc 4 delivery of Inc 4

Calendar time

The first increment is the core product in which basic requirements are addressed but many supplementary features remain undelivered. The core product is used by the customer as a result of use/or evaluation, a plan is developed for the next increment. The planned addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality. This process is repeated following the delivery of each increment, until the complete product is produced.

Incremental model is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project. In addition, increments can be planned to manage technical risks.

Get your ticket follows incremental model because initial software requirements are reasonably well defined but the overall scope of development effort precludes a purely linear process. We use incremental process model, to provide delivery of project earlier. Increments can be planned to manage technical risks.

1. INTRODUCTION
   1. PURPOSE
   2. PROJECT SCOPE
      1. OBJECTIVES OF THE PROPOSED SYSTEM
   3. DEFINITIONS , ACRONYMS , ABBREVIATIONS
   4. REFFERENCES
   5. OVERVIEW

1. INTRODUCTION

A Software (GET YOUR TICKET) has to be developed for automating the manual RAILWAY RESERVATION SYSTEM.

* RESERVE SEATS – Passenger can book a seat whilst sitting at home without any hassle.
* LOGIN – Passenger has to login into the system using id and password.
* REGISTER – Passenger has to register to use the system.
* CANCEL RESERVATION – Passenger can cancel the ticket sitting at home without going to the railway station.
* VIEW RESERVATION STATUS –The passenger needs to enter the PNR number printed on the ticket in order to view their reservation status.
* VIEW HISTORY –The passenger can view their booking history.
* ONLINE PAYMENT –The Passenger can pay online without any hassle using his/her credit/debit/master card.
* BOOK SEAT –The passenger can choose the type of seat they want.

1.1 PURPOSE

The purpose of this project is to describe the railway reservation system which provides the train timing details, reservation, payment and cancellation on various types of reservation namely-

* Confirm Reservation for confirm seat.
* Reservation against Cancellation.
* Waiting list Reservation.
* Online reservation.
* Tatkal Reservation.

The origin of most software systems is the need of a client, who either wants to automate the existing manual systems or desires a new software system. The software system is created by the developer. Finally, the end user will use the completed system. Thus, there are three major parties interested in a new system: the client, the user, and the developer. Somehow, the requirements of the system that will satisfy the needs of the clients and the concerns of the users have to be communicated to the developer. The problem is that the client does not usually design the software or the software development process and the developer does not understand the client’s problem and the application area .This causes a communication gap between the parties involved in the development of the project.

A few factors that direct us to develop a new system are given below :-

1. Faster System
2. Accuracy
3. Reliability
4. Informative
5. Reservations and cancellation from anywhere to any place

1.2 SCOPE

“GET YOUR TICKET” is an attempt to simulate the basic concepts of the online reservation system. The system enables to perform the following functions:

* Search a Train
* Book seat
* Payment
* Cancellation
* View status of booking
* View history
* Update train info
* Improved and optimized service.
* Passenger Revenue enhancement.
* Freight Revenue enhancement

1.2.1 OBJECTIVES OF THE PROPOSED SYSTEM

Our objective is to provide a system which is easy to use , is reliable , secure , as fast response time . Our system is reducing the hassle that is caused by travelling to the railway station to book a ticket or cancel a ticket.

In our system , user can search a train ,book a seat , pay online , cancel a ticket , view status of booking , view booking history .

1.3ABBREVIATIONS, DEFINITION AND ACRONYMS

* Id – identification details
* Ph no. – Phone number
* Info –Information
* DOB – Date of Birth
* DFD – Data Flow Diagram
* ERD – Entity Relationship Diagram
* SRS – Software Requirements Specification

1.4 REFERENCES

[1] R.S. PRESSMAN , SOFTWARE ENGINEERING A PRACTITIONER’S APPROACH , MC. GRAW HILL , ED 7, 2010

[2] P. JALOT , AN INTEGRATED APPROACH TO SOFTWARE , NAROSA PUBLISHING HOUSE , ED . 3, 2011

* 1. OVERVIEW

Our project is based on online railway reservation system.

In this project, our aim is to create a website which will help reduce the time and money wasted for booking rail tickets.

Through our website, the user can book their tickets sitting at home without any hassle of travelling to the railway station and standing in large queues. It reduces wastage in terms of money as cost of travelling to the railway station is reduced, to keep the records of the tickets and everything large amount of paper is required. This makes the ticket booking process fast and reliable.

1. PROJECT DESCRIPTION
   1. PRODUCT PROSPECTIVE
      1. SYSTEM INTERFACES
      2. HARDWARE INTERFACES
      3. COMMUNICATION INTERFACE
      4. MEMORY CONSTRAINTS
      5. OPERATIONS
      6. SITE ADAPTATION REQUIREMENTS
   2. PRODUCT FEATURES
      1. PASSENGERS FUNCTIONALITY
      2. MANAGEMENT TEAM’S FUNCTIONALITY
   3. USER CLASSES AND CHARACTERISTICS
      1. USER CHARACTERISTICS
      2. USER CLASSES
   4. DESIGN AND IMPLEMENTATION CONSTRAINTS
   5. ASSUMPTIONS AND DEPENDENCIES

2. PROJECT DESCRIPTION:

2.1 PRODUCT PERSPECTIVE:

Before the automation, the system suffered from the following

DRAWBACKS:

* The existing system is highly manual involving a lot of paper work and calculation and therefore may be erroneous. This has lead to inconsistency and inaccuracy in the maintenance of data.
* The data which is stored on the paper only, may be lost, stolen on destroyed due to natural calamity like fire and water.
* The existing system is sluggish and consumes a lot of time causing inconvenience to customers.
* Due to manual nature, it is difficult to update, delete, add or view the data.
* Since the number of passengers have drastically increased therefore maintaining and retrieving detailed record of passenger is extremely difficult.
* Railways have many offices around the world, an absence of a link between these offices lead to lack of coordination and communication.
* The users have to go to the railway station to book the tickets, cancel ticket, check their booking status, pay money in cash.

Hence the railways reservation system is proposed with the following:

* The computerization of the reservation system will reduce a lot of paperwork and hence the load on the administrative staff will be reduced.
* The machine performs all calculations. Hence chances of error are almost nil.
* The passenger, reservation, cancellation list can easily be retrieved and any required addition, deletion or updation can be performed.
* The system provides asks for user-ID validation, hence unauthorized access is prevented.

PROJECT FUNCTION:

Booking agents with varying levels of familiarity with computers will mostly use this system. With this in mind, an important feature of this software is that it be relatively simple to use.

The scope of this project encompasses:-

REGISTER:

In order to use this system the user has to register to into this system. The user has to give all the personal details asked by the system such as first name, last name, DOB, address, Ph no., email id, etc. The will have to use a unique user id which will be used to identify every user. The user will also have to enter a password which will be needed by the user every time he/she tries to enter the system.

LOGIN:

In this module, the user will be needed to enter his/her user id and password in order to enter the system. Without loging in the user can not do anything.

SEARCH:

The function allows the booking agent/user to search for train that are available between the two travel cities, namely the “Departure city” and “Arrival city” as desired by the traveler. The system initially prompts the agent/user for the departure and arrival city. It then displays a list of train available with different routes between the designated cities on different dates and time.

BOOK SEAT:

The function allows user to choose the type of seat they want AC or non-AC, chair car or sleeper, vegetarian, non-vegetarian or no food. Then checks the availability of the seat.

ONLINE PAYMENT :

User can pay online using their debit/credit/master card without using cash. Just by entering the CVV number of the card, valid date of the card, card number.

CANCELLATION:

User can cancel their ticket if they want to and their money will be refunded to their bank account.

CHECK STATUS :

User can check the status of their booking by just entering their PNR no.

CHECK BOOKING HISTORY:

User can check the booking that they have made in the past using this module.

UPDATE TRAIN INFO:

The management team can update the train information using this module. All the train information is updated here this can only be done by the management team.

2.1.1 SYSTEM INTERFACE

This system does not have any interfaces with any of the existing

system.

2.1.2 HARDWARE INTERFACE

GET YOUR TICKET uses standard java classes in databases. No additional specific hardware interfaces are needed during the operation of the GET YOUR TICKET website.

2.1.3 COMMUNICATION INTERFACE

No additional specific communication interfaces are needed during the operation of the GET YOUR TICKET website.

2.1.4 MEMORY CONSTRAINTS

The system is expected to have a memory capacity of 256MB and disk space of 500MB. But it is recommended that the system has memory capacity of 1GB and disk space of 1GB.

2.1.5 OPERATION

The basic operations of the GET YOUR TICKET are described below:

* The user when using the website for the first time has to register with the website.
* The user after registering can login to the website with his user name and password.
* The user will be initially shown home page.
* The user can search for a train.
* The user can book a seat.
* The user can pay online using his/her debit/credit/master card.
* The user can check his/her booking status using check status module.
* The user can check his/her booking history.

2.1.6 SITE ADAPTATION REQUIREMENTS

The system will require an application server for the runtime components and a database for storage.

The system will run on select popular application servers and use select popular database for data storage.

* 1. PRODUCT FEATURES

2.2.1 PASSENGER’S FUNCTIONALITY

* Register – New user has to register in order to use this website.
* Login– Registered user will simply login to the software with username and password.
* Search train – User can search for a train by entering the station names.
* Book seat – User can book a seat on this website.
* Payment – User can pay for the ticket online through debit/credit/master card.
* Check Status - User can check status of their booked ticket .
* Booking History – User can see his/her booking history.
* Cancellation- User can cancel the ticket using his/her PNR no.
  + 1. MANAGEMENT TEAM’S FUNCTIONALITY
* Login – Management team has to login to access the website.
* Update Information – Management team can update the database of the train.

2.3 USER CHARACTERISTICS AND CLASSES:

* MANAGEMENT TEAM – Management team logs into the system to update information about train and its schedule. Any change in the train schedule will be uploaded by the management team.
* GENERAL USER – User will have to register into the system in order to use it. User will have to log into the system using his/her id and password. User can search a train, book a seat, pay online, cancel tickets, check status of the tickets using PNR no. and can check booking history.
* PASSENGER- Passenger can register on the website and then login into the website using his/her user id and password. Passenger can search a train, book a seat, pay online, cancel tickets, check status of the tickets using PNR no. and can check booking history.

USER CLASS

There are three types of users:

1. Management team
2. General user(booking agent)
3. Passenger

* 1. DESIGN AND IMPLEMENTATION CONSTRAINTS
* The system will run under windows XP or higher platforms of operating system.
  1. ASSUMPTIONS AND DEPENDENCIES :
* Booking Agent will be having a valid user name and a password to access the software.
* The software needs booking agents to have complete knowledge of online railways reservation system.
* Software is dependent on access to internet.
* It is assumed that the user is familiar with using website.

1. SPECIFIC REQUIREMENTS
   1. EXTERNAL INTERFACE
   2. FUNCTIONAL REQUIREMENTS
      1. PASSENGER’S FUNCTIONALITY
      2. MANAGEMENT TEAM’S FUNCTIONALITY
   3. PERFORMANCE REQUIREMENTS
   4. LOGICAL DATABASE REQUIREMENTS
   5. DESIGN CONSTRAINTS
   6. SECURITY REQUIREMENTS
   7. SOFTWARE SYSTEM ATTRIBUTES
      1. RELIABILITY
      2. AVAILABILITY
      3. SECURITY
      4. MAINTAINABILITY
      5. PORTABILITY
   8. DATA FLOW DIAGRAM
      1. CONTEXT LEVEL DIAGRAM
      2. LEVEL-1 DFD
      3. LEVEL-2 DFD
   9. DATA DICTIONARY
   10. USE CASES
       1. USE CASE DIAGRAM
       2. USE CASE DESCRIPTION
   11. EXTERNAL INTERFACE

Payment gateway is the only external interface identified at this point.

* 1. FUNCTIONAL REQUIREMENTS – These are the services that the system should provide.
     1. PASSENGER’S FUNCTIONALITY
* Register – Input : id, password ,address, first name, last name, DOB, phn no., OTP, email id, security question and answer.

Output: OTP, login screen.

Processing- New user has to register in order to use this website.

* Login– Input- id, password, security answer.

Output- main menu.

Processing-Registered user will simply login to the software with username and password.

* Search train – Input- tatkal or general, station name, train name.

Output-train name, train number, day, dates, railway station, time, platform no.

Processing-User can search for a train by entering the station names.

* Book seat –Input-options.

Output-PNR no., seat no., coach name.

Processing- User can book a seat on this website.

* Payment –

Input-mode, CVV no., valid date, card no.

Output-success message.

Processing-User can pay for the ticket online through debit/credit/master card.

* Check Status -

Input- PNR no.

Output- status of ticket.

Processing-User can check status of their booked ticket .

* Cancellation-

Input-PNR no.

Output-cancellation number, refund amt, train name.

Processing-user can cancel the ticket using this module.

* History-

Input-nil.

Output-past ticket receipts.

Processing-shows the history of the user.

* + 1. MANAGEMENT TEAM’S FUNCTIONALITY
* Login –

Input-id, password.

Output-main menu.

Processing-Management team has to login to access the website.

* Update train info-

Input-train details.

Output-nil.

Processing- management team uses this module to update train info.

* 1. PERFORMANCE REQUIREMENTS
* User Satisfaction – The system is such that it stands up to the user expectations.
* Response Time – The response of all the operation’s good. This has been made possible by careful programming.
* Error Handling – Response to user errors and undesired situation has been taken care of to ensure that the system operate without halting.
* Safety and Robustness – The system is able to avoid or tackle disastrous action. In other words, it should be foul proof. The system safeguards against undesired events, without human intervention.
* Portable – The software should not be architecture specific. It should be easily transferable to other platforms if needed.
* User friendliness – The system is easy to learn and understand. A native user can also use the system effectively, without any difficulties.
  1. LOGICAL DATABASE REQUIREMENTS

The database contains the following informations:

* The registration details.
* The details entered by the user.
* Details of train schedule.
* Payment details
* Ticket details
* Cancellation details
  1. DESIGN CONSTRAINTS

SOFTWARE CONSTRAINTS

* GET YOUR TICKET website shall be designed for future upgrade as per needed.

.

HARDWARE CONSTRAINTS

The system has memory capacity of 1GB and disk space of 1GB. This website can be accessed through a computer and also a mobile. The system will run under windows 7 or higher platforms of operating system. SQL is supported by this system.

* + 1. STANDARD COMPLIANCE

Report format: All the reports produced for this project are in compliance with the standard templates provided in the class by the advisors.

Naming format: All the documents will be named using the standard naming conventions.

* 1. SECURITY REQUIREMENTS

The data entered by the user such as personal details will be kept confidential. No user will be able to access other user’s personal details. The web application shall be secure from hackers and other malware. The system backend servers can only be accessible to authenticated management.

* 1. SOFTWARE SYSTEM ATTRIBUTES
     1. RELIABILITY

The reliability of the overall project depends on the reliability of the separate components. The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated to reflect the most recent changes. Also the system will be functioning inside a container .Thus the overall stability of the system depends on the stability of container and its underlying operating system.

* + 1. AVAILABILITY

The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs. A user friendly system which is in access of people around the world should work 24 hours. In case of a hardware failure or database corruption, a replacement page will be shown. Also in case of a hardware failure or database corruption, backups of the database should be retrieved from the server and saved by the organizer .Then the service will be restarted. It means 24X7 availability.

* + 1. SECURITY

The data entered by the user such as personal details will be kept confidential. No user will be able to access other user’s personal details. The web application shall be secure from hackers and other malware. The system backend servers can only be accessible to authenticated management

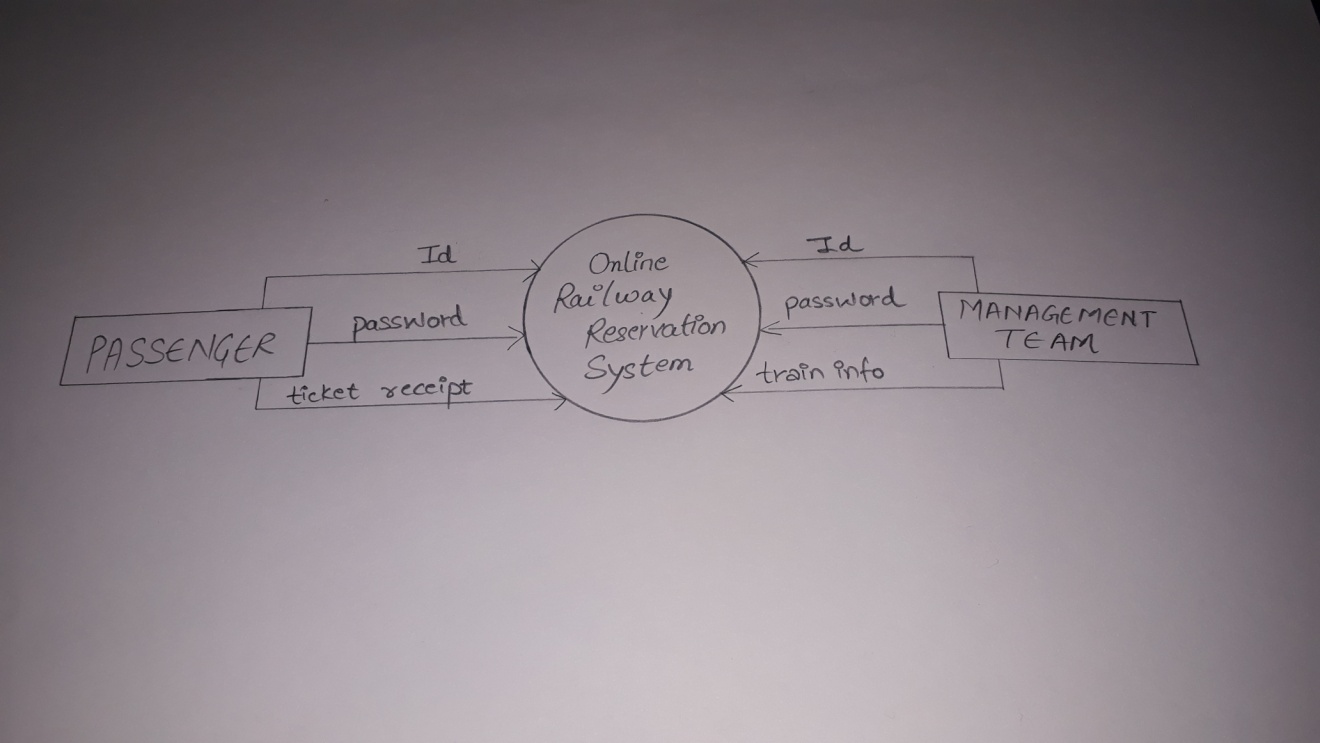
* + 1. MAINTAINABILITY

A commercial database is used for maintaining the database and the application server takes care of the site. In case of a failure, a re-initialization of the project will be done . Also the software design is being done with modularity in mind so that maintainability can be done efficiently.

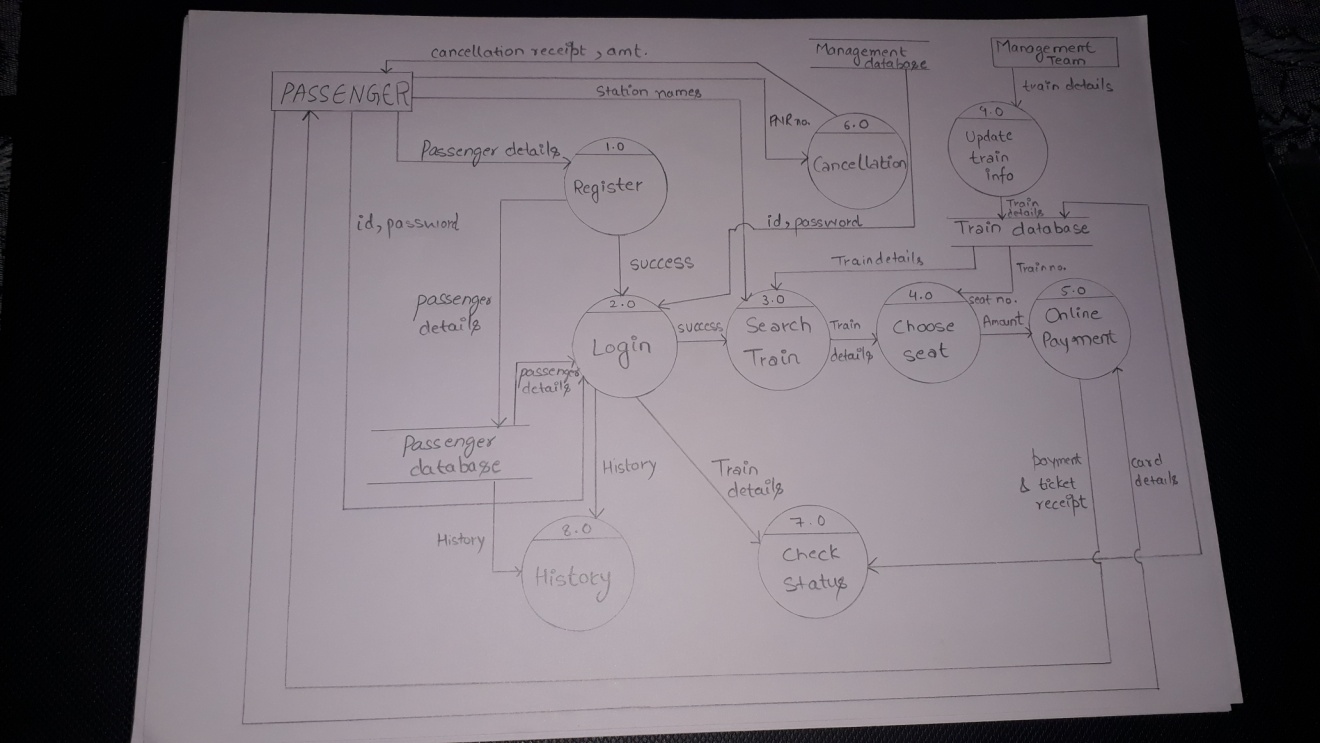
* + 1. PORTABILITY

No portability requirements are identified at this time.

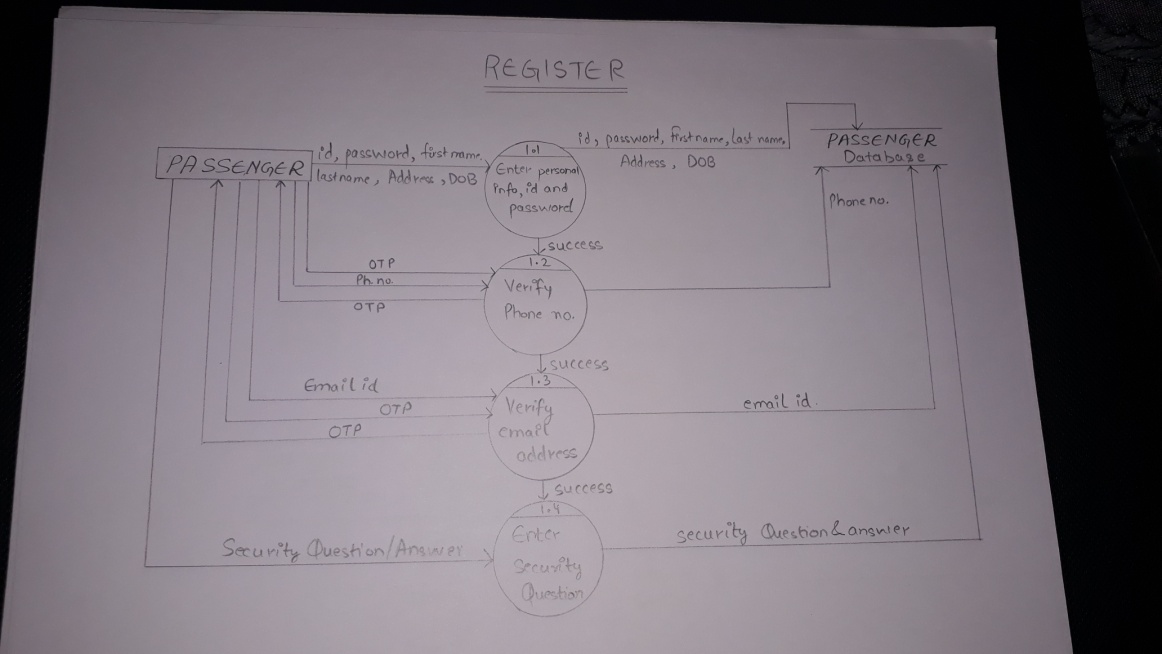
* 1. DATA FLOW DIAGRAM
     1. CONTEXT LEVEL DIAGRAM



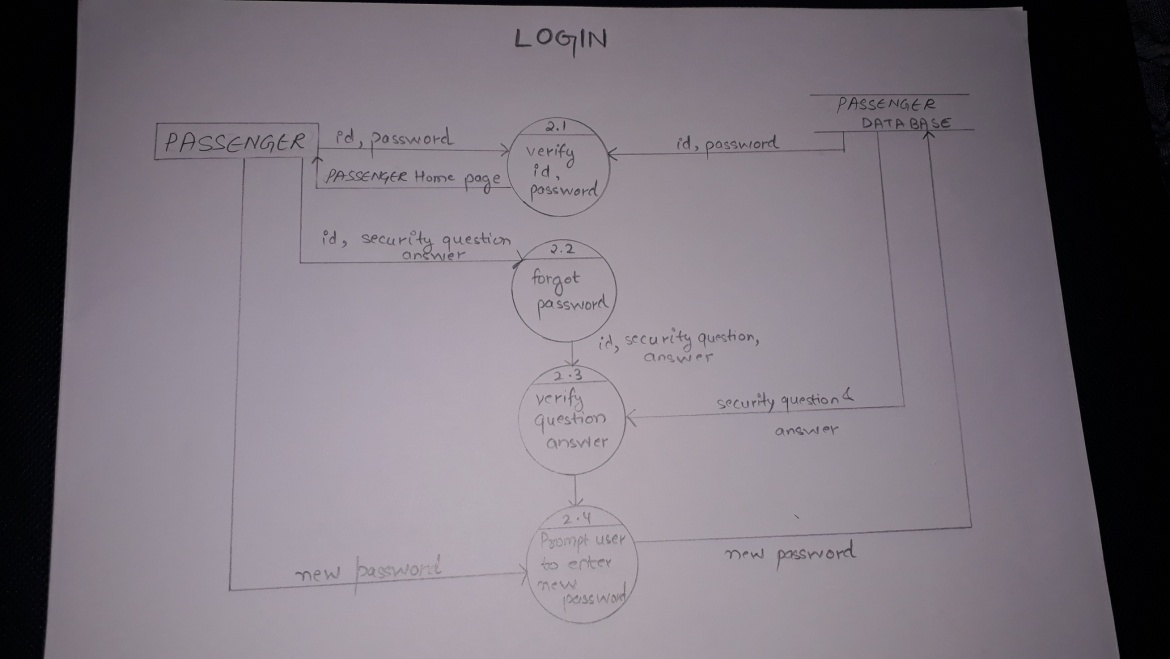
* + 1. LEVEL-1 DFD



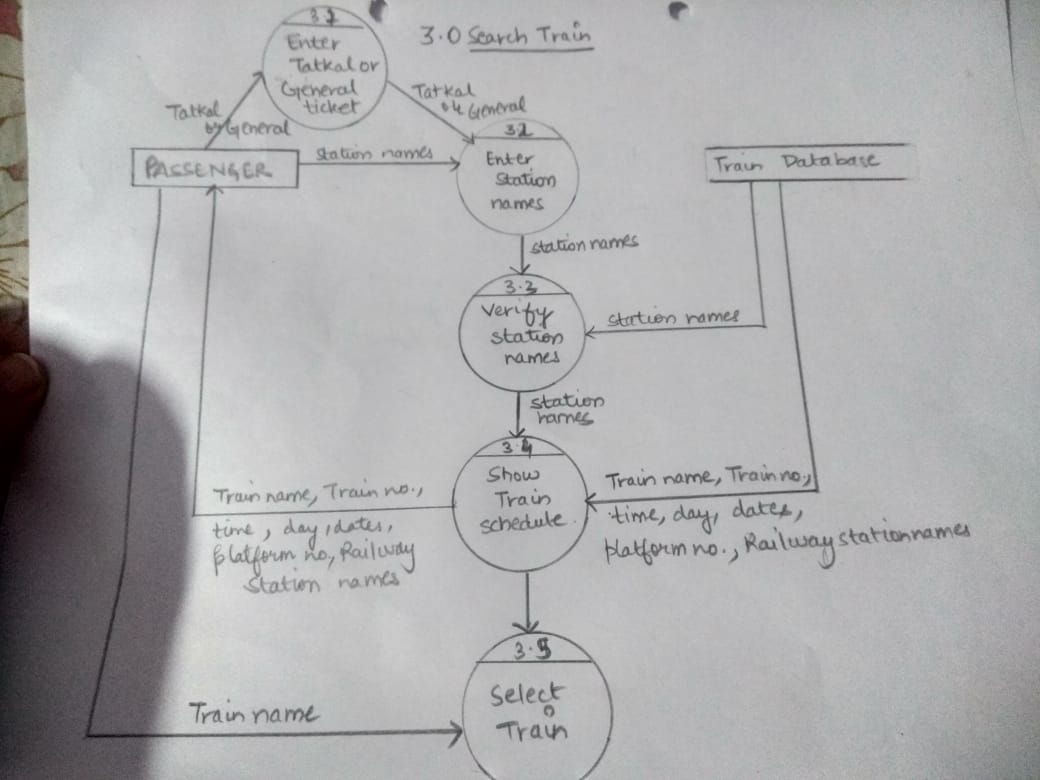
* + 1. LEVEL-2 DFD (REGISTER)



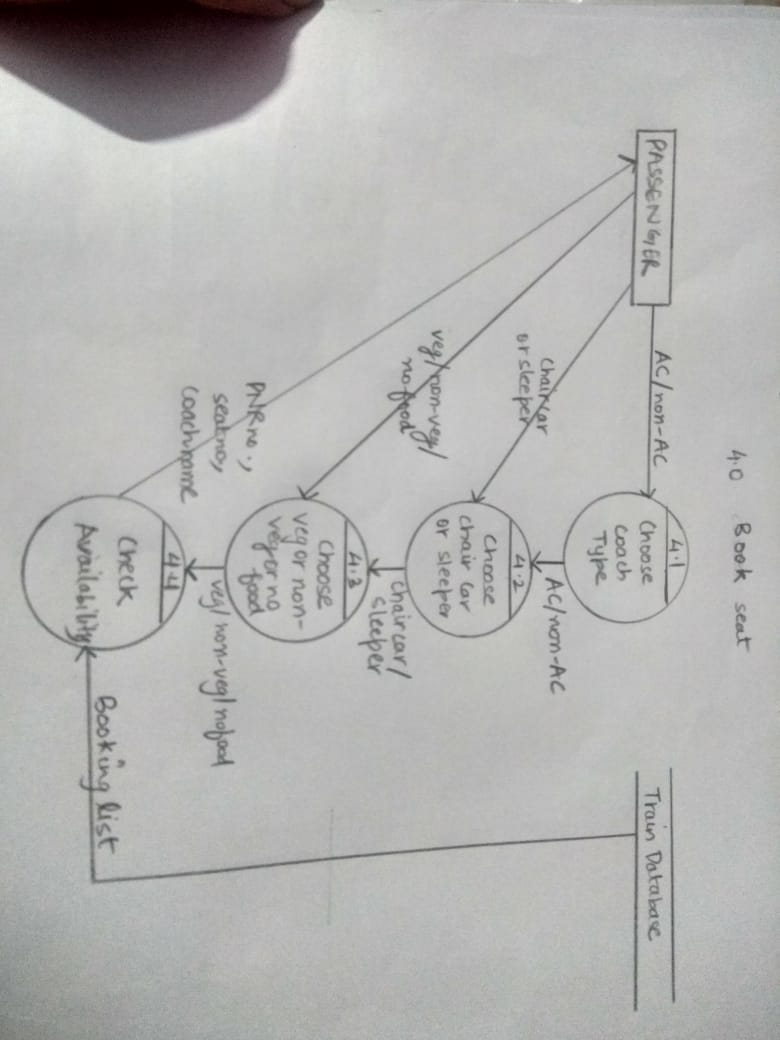
LEVEL-2 DFD (LOGIN)



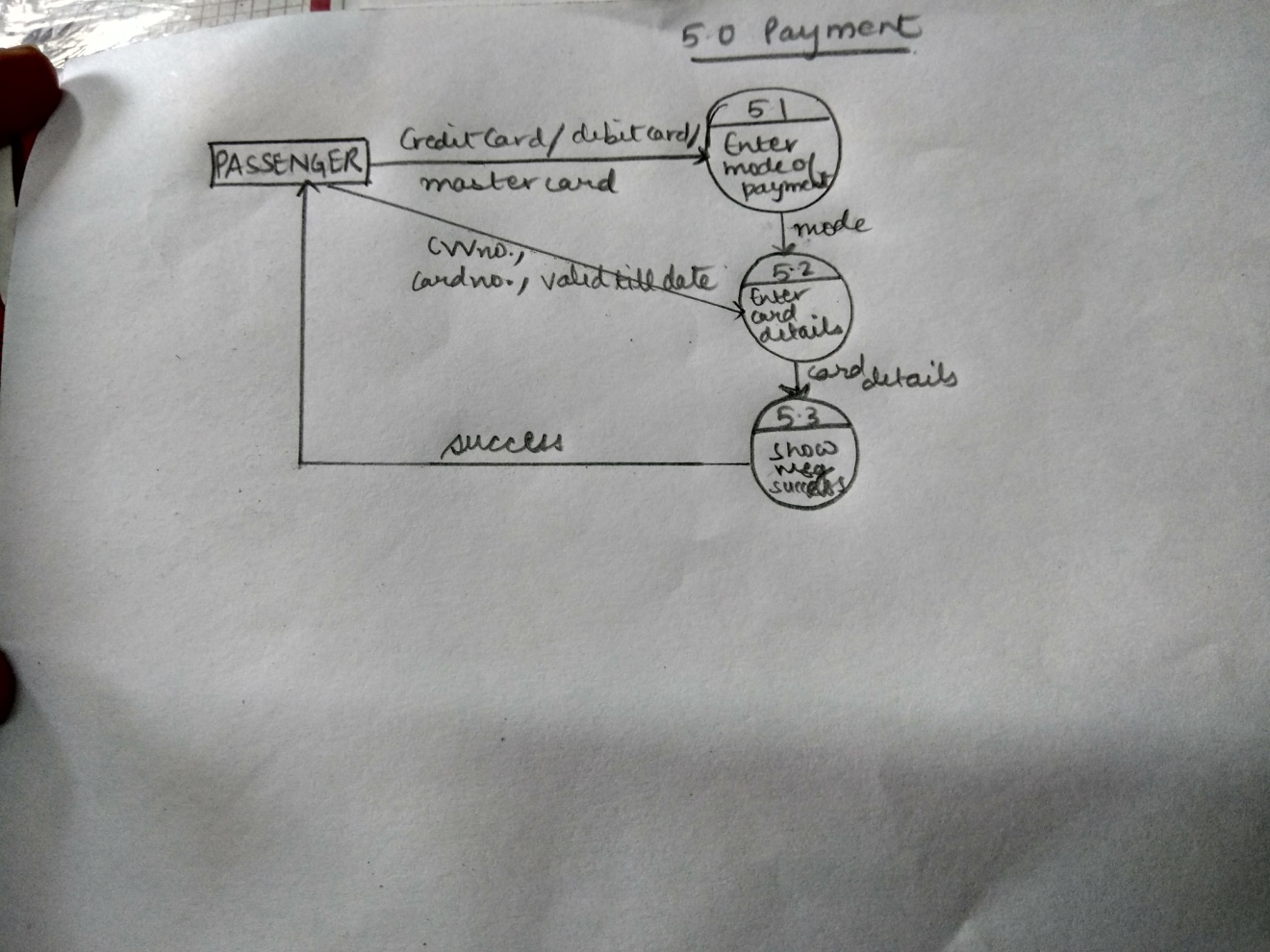
LEVEL-2 DFD (SEARCH TRAIN)



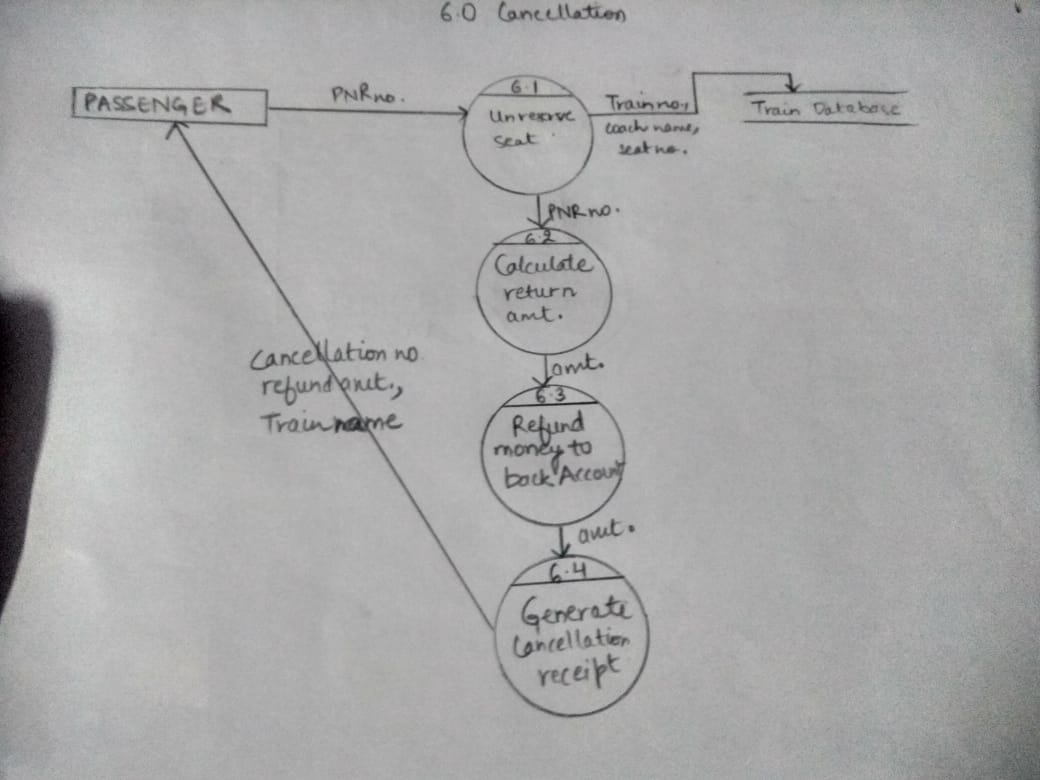
LEVEL-2 DFD (BOOK SEAT)



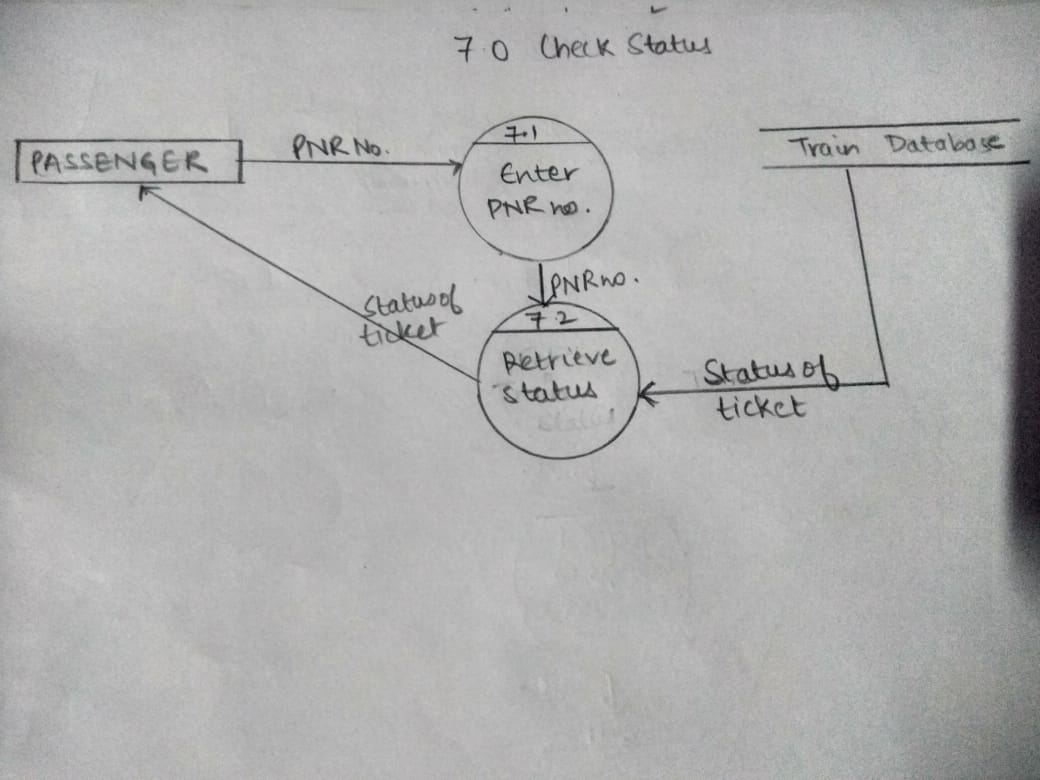
LEVEL-2 DFD (ONLINE PAYMENT)



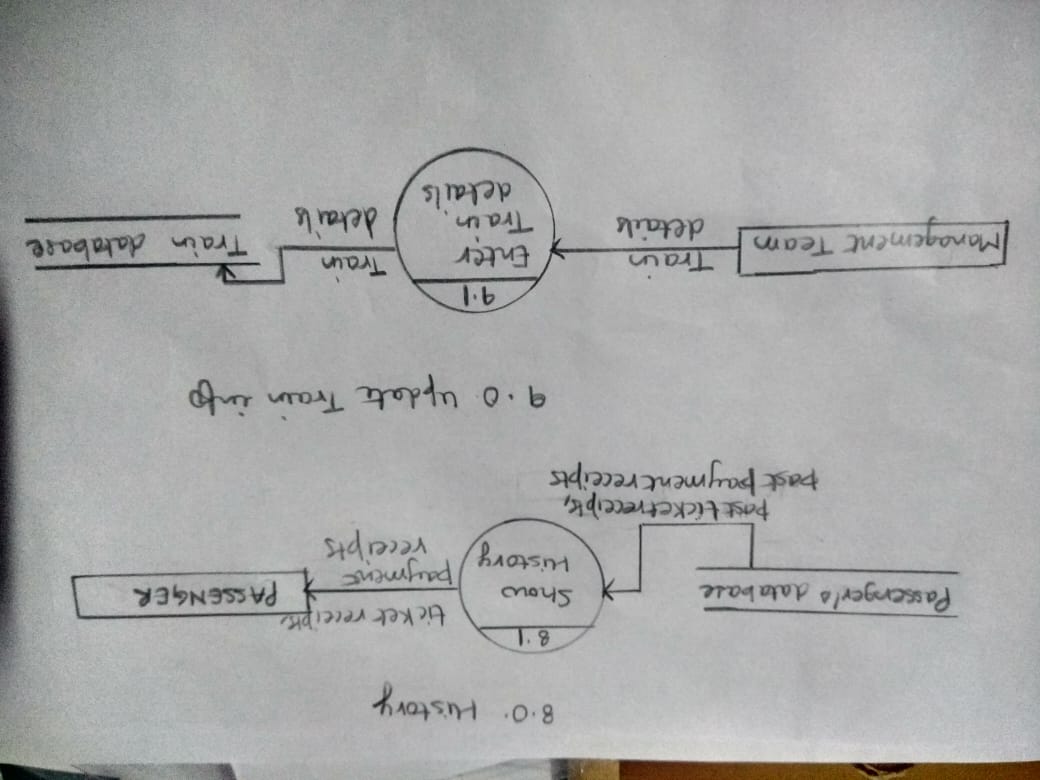
LEVEL-2 DFD (CANCELLATION)



LEVEL-2 DFD (CHECK STATUS)



LEVEL-2 DFD (HISTORT AND UPDATE TRAIN INFO.)



* 1. DATA DICTIONARY

Id ={legal character}\*

Password = {legal character}\*

Train info = { train name + train no + arrival time + departure time + source + destination +date }

Ticket receipt = {PNR no + Ticket no + amount}

Ticket no = {digit + digit + digit + digit + digit}

Train details = {train name + train no + arrival time + departure time + source + destination +date}

Payment receipt = {legal character}\*

PNR no. = {digit + digit + digit + digit + digit + digit + digit + digit + digit + digit}

Status of ticket =

Train name= {charac}\*

Train \_no = {digit}\*

Arrival time= {digit + digit + : +digit +digit}

Departure time= {digit + digit + : +digit +digit}

source = {charac}\*

destination = {charac}\*

date= {digit + digit + / + digit + digit + / + digit + digit + digit + digit}

cancellation no ={digit + digit + digit + digit}

train name={charac}\*

refund amount = {digit}\*

seat no ={digit + digit + digit}

coach name = {charac}\*

Quota type= {charac}\*

Card details = {CVV no + card no + valid till }

CVV no. ={digit + digit + digit}

Card no. = { digit + digit + digit + digit + digit + digit + digit + digit + digit + digit + digit + digit + digit + digit + digit + digit }

Amount ={digit}\*

Amt. = {digit}\*

OTP = {digit + digit + digit + digit + digit}

Booking list = {no. of passengers}

No. of passengers = {digits}\*

Station names = {legal character}\*

Platform no. = {legal character}\*

station details ={ station name }

Security question = {legal character}\*

Security answer = {legal character}\*

Ph no. = {digit + digit + digit + digit + digit + digit + digit + digit + digit + digit }

Email id = {legal character}\*

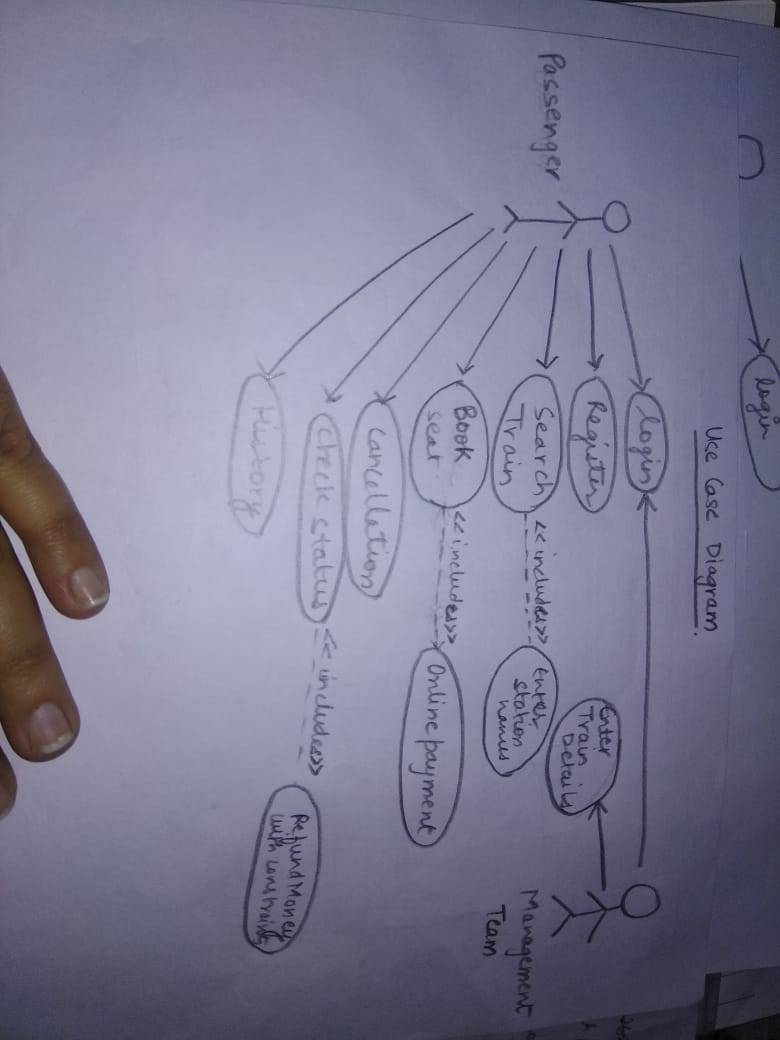
Address={legal character}\*

Legal character = [0-9|a-z|A-Z|\_|@|$|#|+|-]

Digit= [0-9]

Charac=[A-Z|a-z]

* 1. USE CASES
     1. USE CASE DIAGRAM



* + 1. USE CASE DESCRIPTION

❖**Login**

**1. Brief Description**

This use case describes how an actor logs into the ‘GET YOUR TICKET’ system.

**2. Actors**

USER

PASSENGER

MANAGEMENT TEAM

**3. Flow of events**

1. a) Basic flow

This use case starts when the actor wishes to log in to the GET YOUR TICKET system.

The system requests that the actor enter his/her ID and password.

The actor enters his/her ID and password.

The system validates the entered ID and password and the actor is then logged in to the system.

1. b) Alternative flows

If in the basic flow, the actor enters an invalid ID or password, the system displays an error message. The actor can use to either return to the beginning of the basic flow or cancel the login at the point where use case ends.

**4. Special Requirements**

None

**5. Pre-conditions**

All actors must have an account created for them in the system prior to executing the use cases.

**6. Post conditions**

If the use case was successful, the actor is logged in to the system. If not, the system state is unchanged.

Every actor has the access to the corresponding screens to his/her role.

**7. Extension points**

None

❖**Register**

1.**Brief Description**

This use case describes how an actor registers into the ‘GET YOUR TICKET’ system.

**2. Actors**

User

Passenger

**3. Flow of events**

1. a) Basic flow

This use case starts when the actor wishes to register to the GET YOUR TICKET system.

The system requests that the actor enter his/her first name, last name, security question ,security answer, phone number, DOB, , address, email, nationality , occupation , id and password.

The actor enters his/her first name, last name, security question ,security answer, phone number, DOB, , address, email, nationality , occupation , id and password.

The system stores the entered attributes in the database and the actor is then registered to the system. S/He then needs to enter only id to log in to the system again.

The system displays a confirmation message on the screen.

**4. Special Requirements**

None

**5. Pre-conditions**

None

**6. Post conditions**

If the use case was successful, the actor is registered to the system. If not, the system state is unchanged.

**7. Extension points**

None

❖**search train**

1. **Brief Description**

This use case describes how an actor can search a for a train.

2.**Actors**

User

Passenger

**3. Flow of events**

1. a) Basic flow

This use case starts when the actor wishes to search a train. The user has to enter a valid source and destination station. Then the system shows the schedule of the trains that travel between these destinations.

**4. Special Requirements**

None

**5. Pre-conditions**

The must have registered prior this use case.

**6. Post conditions**

If the use case was successful, the system shows the train schedule. Actor can now book a seat after this use case is completed.

**7. Extension points**

None

**BOOK SEAT**

**1.Brief Description**

This use case describes how an actor can book a seat for themselves.

**2. Actors**

User

Passenger

**3. Flow of events**

1. a) Basic flow
2. The user should have choose a train before this module so that the system look for the availability of the seats for that particular train.

**4. Special Requirements**

None

**5. Pre-conditions**

All actors must have an account created for them in the system prior to executing the use cases.

**6. Post conditions**

If the use case was successful, the use case takes the user to payment use case.

**7. Extension points**

None

**Payment**

1. **Brief Description**

This use case describes how the user can pay.

**2. Actors**

User

Passenger

**3. Flow of events**

1. a) Basic flow

This use case starts when the actor needs to pay online for the tickets.

**4. Special Requirements**

The actors must have a card to pay online.

**5. Pre-conditions**

The actor must have a debit/credit/master card to pay online.

**6. Post conditions**

If the use case was successful, money is deducted from the actor’s account.

**7. Extension points**

None

❖**cancellation**

1.**Brief Description**

This use case describes how an actor approves the invoice shown to the doctor of the payment made to him/her.

**2. Actors**

User

Passenger

**3. Flow of events**

1. a) Basic flow

This use case starts when the actor needs to cancel the ticket of the train .

**4. Special Requirements**

None

**5. Pre-conditions**

None

6**. Post conditions**

If the use case was successful, the ticket will be cancelled. If not, the system state is unchanged.

**7. Extension points**

None

❖**Check Status**

1.**Brief Description**

This use case describes how an actor checks the booking status of his/her train.

**2. Actors**

User

Passenger

**3. Flow of events**

1. a) Basic flow

This use case starts when the actor needs to check the status of the train to know whether the seat is reserved or not.

The system requires the actor to PNR no.

**4. Special Requirements**

None

**5. Pre-conditions**

None

**6. Post conditions**

If the use case was successful, the system will show the status . If not, the system state is unchanged.

**7. Extension points**

None

❖**History**

1.**Brief Description**

This use case describes how an actor check the booking history of the seats.

**2. Actors**

User

Passenger

3**. Flow of events**

1. a) Basic flow

This use case starts when the actor needs to check the booking history .

**4. Special Requirements**

None

**5. Pre-conditions**

None

**6. Post conditions**

If the use case was successful, the system will show the booking history . If not, the system state is unchanged.

**7. Extension points**

None

❖**Update train info.**

1.**Brief Description**

This use case describes how an actor update the train info that is how many seats are left unreserved and all the details of the train .

**2. Actors**

Management team

3**. Flow of events**

1. a) Basic flow

This use case starts when the actor needs to update the train info.

**4. Special Requirements**

None

**5. Pre-conditions**

None

**6. Post conditions**

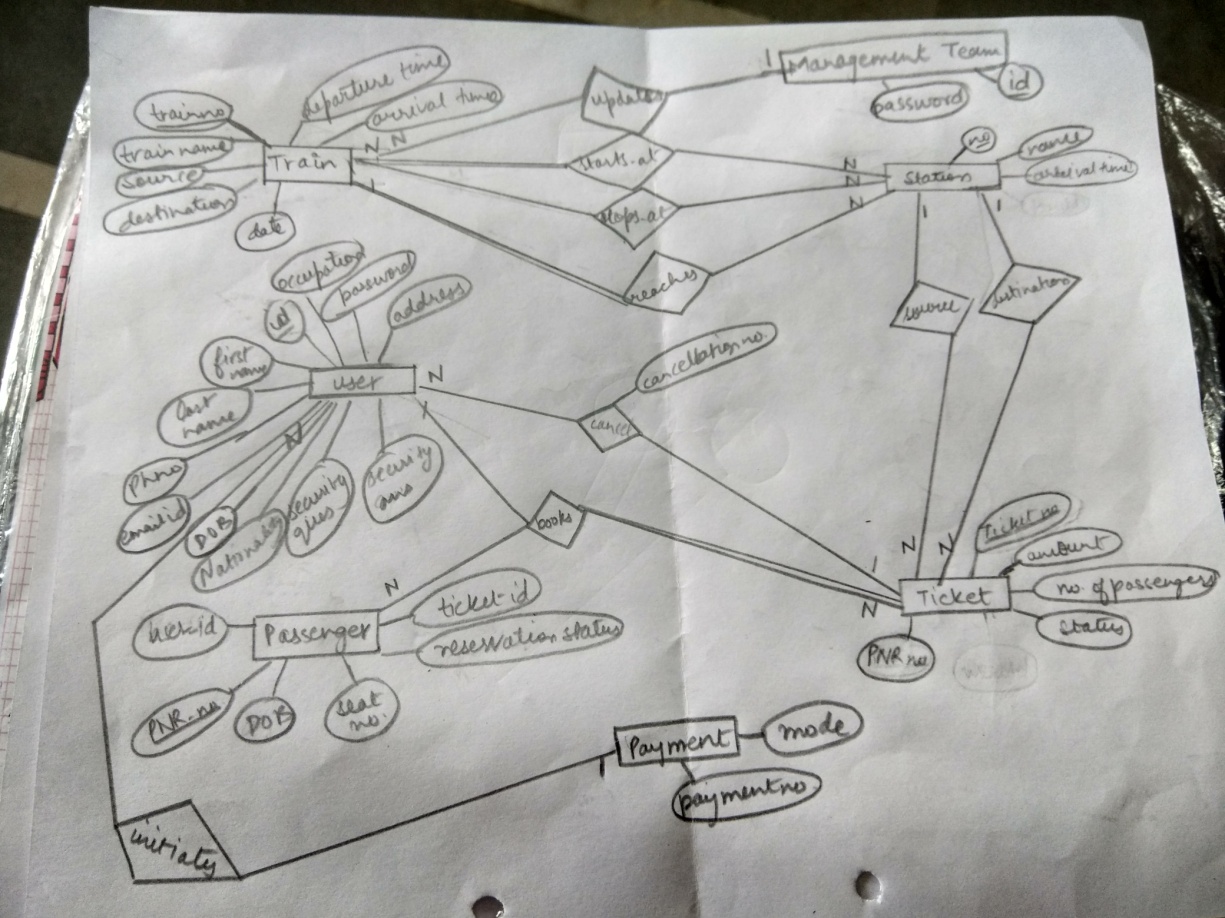
If the use case was successful, the system will update the train info . If not, the system state is unchanged.

**7. Extension points**

None

1. DESIGN
   1. ER DIAGRAM
   2. DATA DESIGN
   3. COMPONENT LEVEL DESIGN

ER Diagram



DATA DESIGN

Management Team

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S no. | FILE NAME | DATA TYPE | FIELD LENGTH | CONSTRAINTS | DESCRIPTION |
| 1. | Id | Alphanumeric | 15 | Primary key | Gives id of the management team |
| 2. | Password | Alphanumeric | 10 | Not null | Used to login into the software |

User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sno. | FIELD NAME | DATA TYPE | FIELD LENGTH | CONSTRAINTS | DESCRIPTION |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  12. | Id  Password  Phone no.  Nationality  Address  First name  Last name  DOB  Security question  Security  Answer  Email id | Alphanumeric  Alphanumeric  numeric  Alphanumeric  Alphanumeric  Alphanumeric  Alphanumeric  Numeric  Alphanumeric  Alphanumeric  Alphanumeric | 15  10  10  20  50  20  20  10  150  150  50 | Primary key  Not null  Not null  Not null  Not null  Not null  Not null  Not null  Not null  Not null  Not null | Gives the id of the customer  Used to login into the software  Gives the contacts of the  Customer  Gives the nationality of the customer  Gives the address of the customer  Gives the first name of the customer  Gives the last name of the customer  Gives the DOB of the user  Gives the security questions of the user    Gives the security answer of the user  Gives the email id of the user |

Train

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SNO. | FIELD NAME | DATA TYPE | FIELD LENGTH | CONSTRAINTS | DESCRIPTION |
| 1.  2.  3.  4.  5.  6.  7. | Ticket no.  Train name  Departure Time  Arrival Time  Date  Destination  Source | Numeric  Alphanumeric  Numeric  Numeric  Numeric  Alphanumeric  Alphanumeric | 20  20  10  10  10  20  20 | Not null  Not null  Not null  Not null  Not null  Not null  Not null | Ticket number on the receipt  Train name  Departure Time of the train  Specifies the Arrival time  Specifies the date on which the journey starts  Destination of the train  Name of the Source station |

Cancel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SNO. | FIELD NAME | DATA TYPE | FIELD LENGTH | CONSTRAINTS | DESCRIPTION |
| 1. | Cancellation no. | Numeric | 10 | Not Null | Cancellation number of the receipt |

Payment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SNO. | FIELD NAME | DATA TYPE | FIELD LENGTH | CONSTRAINTS | DESCRIPTION |
| 1. | payment no. | Numeric | 10 | Not null | Payment number |
| 2. | Mode | Alphanumeric | 20 | Not null | Which mode(credit card, debit card ,master card) |

Passenger

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SNO. | FIELD NAME | DATA TYPE | FIELD LENGTH | CONSTRAINTS | DESCRIPTION |
| 1. | user\_id | Alphanumeric | 10 | Not null | User identification of the passenger or the user |
| 2. | PNR\_no | Numeric | 10 | Not null | PNR number of the receipt |
| 3. | Ticket\_no | Numeric | 10 | Not null | Ticket number |
| 4. | Reservation status | Alphanumeric | 10 | Not null | Status of the reservation |
| 5. | DOB | Numeric | 10 | Not null | Date of birth of the passenger |
| 6. | Seat no | Numeric | 10 | Not null | Seat number |

Ticket

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| SNO. | FIELD NAME | DATA TYPE | FIELD LENGTH | CONSTRAINTS | DESCRIPTION |
| 1. | Ticket no. | Numeric | 10 | Not null | Ticket number |
| 2. | Amount | Numeric | 10 | Not null | Amount to be paid |
| 3. | No.of passenger | Numeric | 500 | Not null | Number of passengers |
| 4. | Status | Alphanumeric | 20 | Not null | Booking status |
| 5. | PNR no. | Numeric | 10 | Not null | PNR number |

**Component level diagram**

Cancellation(int PNR\_no.)

if(pnr\_no\_exists(PNR\_no.)

int days,amt

unreserved\_seat(PNR\_no.)

days=calculate\_days\_left(PNR\_no.)

if(quota\_exixts(PNR\_no.)

amt=cal\_quota\_amt(PNR\_no.)

else amt=amtofticket(PNR\_no.)

endif

rfd\_amt\_to\_bank(Re\_amt)

cncl\_no=gnrate\_cncl\_no(PNR\_no.)

cncl\_receipt=generate\_cacl\_receipt(cncl\_no.,re\_amt,get\_trn\_name

(PNR\_no.))

else print “ENTER A VALID PNR NUMBER”

Endif

end cancellation

1. ESTIMATION AND SCHEDULING
   1. PROJECT SCHEDULING
   2. TIMELINE CHART
   3. SIZE ESTIMATION (FUNCTION BASED METRICES)
   4. COST ESTIMATION (COCOMO II MODEL)

**PROJECT SCHEDULING**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Work tasks** | **Planned work** | **Actual start** | **Planned complete** | **Actual complete** | **Assigned person** | **Effort allocate** |
| **Problem Statement** | Jan W1 | Jan W2 | Jan W3 | Jan W3 | Vanika , Sukhleen | 2 person per week |
| **S/W lifecycle model** | Jan W2 | Jan W3 | Jan W3 | Jan W3 | Vanika | 1 person per week |
| **SRS** | Feb W2 | Feb W2 | March W2 | March W2 | Vanika , Sukhleen | 2 Person per week |
| **Entity Relationship diagram** | March W1 | March W2 | March W2 | March W2 | Vanika , Sukhleen | 2 Person per week |
| **Data Dictionary** | March W2 | March W2 | March W3 | March W3 | Sukhleen | 1 Person per week |
| **Context level Diagram** | March W2 | March W2 | March W2 | March W2 | Sukhleen , Vanika | 2 Person per week |
| **DFD 1** | March W2 | March W2 | April W2 | April W2 | Vanika , Sukhleen | 2 Person per week |
| **DFD 2** | March W3 | March W3 | April W2 | April W2 | Vanika , Sukhleen | 2 Person per week |
| **Use Case Diagram** | March W2 | March W2 | March W3 | March W3 | Vanika | 1 Person per week |
| **Use Case Description** | March W3 | March W3 | March W4 | March W4 | Sukhleen | 1 Person per week |
| **Function Point Metrics** | March W4 | March  W4 | April  W1 | April  W1 | Sukhleen , vanika | 1 Person per week |
| **Effort**  **Estimation**  **Using COCOMO**  **model** | April W2 | April  W2 | April  W2 | April  W2 | Vanika | 1 Person per week |
| **Risk Analysis** | April W2 | April W2 | April W2 | April W2 | Sukhleen | 1 Person per week |
| **Testing** | April W1 | April W1 | April  W3 | April  W3 | Vanika , Sukhleen | 1 Person per week |

**TIMELINE CHART**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **January**  **Work Tasks** | **W1** | **W2** | **W3** | **February**  **W4** | **W1** | **W2** | **W3** | **March**  **W4** | **W1** | **W2** | **W3** | **W4** | **April**  **W1** | **W2** | **W3** | **W4** |
| **Problem Statement** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Software Lifecycle Model** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Project Scheduling** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Software Requirement Specification** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Entity Relationship Diagram ERD** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Data Dictionary** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Context Level Diagram** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **DFD1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **DFD 2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Use case diagram** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Use case description** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Function point Metrics** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Estimation COCOMO** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Risk Analysis** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **Testing** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**PROJECT METRICS**

Project metrics are used to control and coordinate software engineering process and to improve quality of the software to be produced. Project specific metrics provide indication of productivity and insight into the technical activities.

FUNCTION ORIENTED METRICS

Function oriented metrics use function point as normalization value. Function points are derived using empirical relationship based on countable measure of software’s information domain and qualitative assessments of software complexity.

To compute function points (FP), the following relationship is used:

FP=count total \* [0.65 + 0.05\* ∑(fi)]

Where count total is the sum of all FP entries obtained from the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Information domain value | Count | simple | Weighing factor  Average | complex |  |
| External inputs | 58X | 3 | 4 | 6 | 174 |
| External outputs | 26X | 4 | 5 | 7 | 104 |
| External Inquiries | 2X | 3 | 4 | 6 | 6 |
| Internal logical files | 9X | 7 | 10 | 15 | 91 |
| External interface files | 1X | 5 | 7 | 10 | 5 |
| Count total |  |  |  |  | 380 |

The fi (i=1 to 14) are value adjustment factors (VAF) based on following responses:

|  |  |  |
| --- | --- | --- |
| 1. | Does the system require reliable backup and recovery? | 3 |
| 2. | Are specialized data communications required to transfer information to or from the website? | 2 |
| 3. | Are there distributed processing functions? | 3 |
| 4. | Is performing critical? | 4 |
| 5. | Will the system run in an existing, heavily utilized operational environment? | 4 |
| 6. | Does the system require online data entry? | 5 |
| 7. | Does the online data entry require the input transaction to be built over multiple screens or operations? | 5 |
| 8. | Are the IFLs updated online? | 5 |
| 9. | Are the inputs, outputs, files, or inquiries complex? | 1 |
| 10. | Is the internal processing complex? | 1 |
| 11. | Is the code designed to be reusable? | 3 |
| 12. | Are conversion and installation included in the design? | 0 |
| 13. | Is the system designed for multiple installations in different organizations? | 3 |
| 14. | Is the website designed to facilitate change and ease of use by the user? | 3 |
|  | ∑fi | 42 |

Therefore,

FP = 380\*[0.65+(0.01\*42)]

=406.6

EFFORT ESTIMATION USING COCOMO MODEL

COnstructive COst MOdel (COCOMO II)is a more comprehensive estimation model. COCOMO II is actually a hierarchy of estimation models that address the following ares:

* Application composition model – Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
* Early design stage model – Used once requirements have been stabilized and basic software architecture has been established.
* Post-architecture-stage mode – Used during the construction of the software.

The COCOMO II models require sizing information. Three different sizing options are available as part of the model hierarchy:

* Object points
* Function points
* Lines of source code

The object point is an indirect software measure that is computed using counts of the number of (1) screens (at the user interface), (2) reports and (3)components likely to be required to bring the website each object instance is classified into one of three complexity levels based on the following table –

|  |  |  |  |
| --- | --- | --- | --- |
| Object type | Simple | Complexity weight  Medium | Difficult |
| Screens | 1 | 2 | 3 |
| Reports | 2 | 5 | 8 |
| 3GL |  |  | 10 |

Figure – complexity weighting for object types.

The objects count is determined by multiplying the total number of object instances by weighting factor.

When component based development or general software re-used is to applied, the percent of re-use is estimated and object count is adjusted:

NOP = (object points) X [(100-%re-use)/100]

Where NOP is defined as new object points

To derive an estimation of effort based on the computed NOP value , a “productivity rate” must be derived.

PROD =NOP/ person-month

Figure – Productivity rate for object points

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Developer’s experience capability | Very low | Low | Normal | High | Very high |
| Environment maturity capability | Very low | Low | Normal | High | Very high |
| PROD | 4 | 7 | 13 | 25 | 50 |

Once the productivity rate has been determined , an estimated of project effort is computed using :-

Estimated effort =NOP/PROD

COCOMO Estimation for our project –

Number of screens = 19

Number of reports = 2

Number of 3GL components used = 0

In our project, there are simple screens and reports.

So, Object point = 19\*2+2\*5

=38+10=48

Since we’re not re-using any of the components in our project, the % re-use is zero here.

NOP = 48\*[(100-0)/100]

=48

Assuming, the average Productivity for this kind of system = 6.5 FP/pm

Now,

Considering the labour rate = RS 8000

Total Estimated Efforts = 48/6.5=7.3846 person-month

Total cost for the project = Total efforts \* labour rate

=7.3896 \* 8000

=59076.923

1. RISK ANALYSIS

**Risk identification :**

Product Size – The customer’s system should have the capability to fulfill the system’s size.

Business impact – No risk associated with constraints imposed by management.

Stakeholder Characteristics – No risk associated with the stakeholder and developer’s ability to communicate with the stakeholders in timely.

Process definition – The process is well defined and is followed by development organization.

Development Environment – No risk associated with the availability and quality of the tools to be used to built the product .

Staff size and experience – The staff size is appropriate for the project and the members are very experienced in their own field. They all are comfortable with the technology.

RISK TABLE

|  |  |  |  |
| --- | --- | --- | --- |
| RISKS | CATEGORY | PROBABILITY | IMPACT |
| Size estimation may be significantly low  Large number of users than planned  Less reuse than planned  End user resist system  Delivery deadline  Will be tightened  Funding will be lost  Technology will not meet expectations  Staff turnover will be high | PS  PS  PS  BU  BU  CU  TE  ST | 40%  60%  20%  10%  50%  20%  10%  50% | Critical  Marginal  Critical  Marginal  Critical  Catastrophic  Catastrophic  Critical |

PROJECT RISK – The risks that threaten the project plan . That is , if project risks become real , it is likely that the project schedule will slip and that costs will increase.

TECHNICAL RISKS – The risks that threaten the quality and timeliness of the software to be produced .If a technical risk becomes a reality , implementation may become difficult or impossible .

RISK MITIGATION , MONITORING AND MANAGEMENT

MITIGATION

The cost of the project would rise up too much if the requirements are changed after the subsequent steps have commenced. To mitigate the risk we would want to:

* Make sure that the data collected is authentic and as extensive as possible.
* Put forward a deadline for proposing changes after which changes would be levied for each subsequent change proposed

MONITORING

While working on SRS , we should conduct multiple reviews to make sure the requirements are understood well and does not have to be changed much later.

MANAGEMENT

In case there is no other option but to make a change in the SRS , the development team must cease their work until the change in the requirements is done and negotiated. Since incremental model is used the changes can be updated in the next increment if the changes are not urgently required by the stakeholder.

Assessing Overall Project Risk :

1. Have top software and customers ,managers formally committed to support the project ?

YES

1. Are end users enthusiastically committed to the project and the system/product to be built?

YES

1. Are requirements fully understood by software engineering team and its customers ?

YES

1. Have customers been involved fully in the definition of requirements ?

YES

1. Do end users have realistic expectations ?

YES

1. Is the project’s scope stable ?

YES

1. Does the software engineering team have the right mix of skills ?

YES

1. Are projects requirements stable ?

YES

1. Does the project team have experience with the technology to be implemented ?

YES

1. Is the number of people on the project team adequate to do job ?

YES

1. Do all customers / user constituencies agrees on the importance of the project and the requirements for the system / product to be built ?

YES

1. TESTING
   1. BASIS PATH SET
   2. CYCLOMATIC COMPLEXITY
   3. CONTROL FLOW GRAPH

7.1 BASIS PATH SET

We will be using the cancellation module for testing:

Cancellation(int PNR\_no.)

1. if(pnr\_no\_exists(PNR\_no.)
2. int days,amt

unreserved\_seat(PNR\_no.)

days=calculate\_days\_left(PNR\_no.)

3. if(quota\_exixts(PNR\_no.)

4. amt=cal\_quota\_amt(PNR\_no.)

5. else amt=amtofticket(PNR\_no.)

6. endif

7. rfd\_amt\_to\_bank(Re\_amt)

8. cncl\_no=gnrate\_cncl\_no(PNR\_no.)

9. cncl\_receipt=generate\_cacl\_receipt(cncl\_no.,re\_amt,get\_trn\_name

(PNR\_no.))

10. else print “ENTER A VALID PNR NUMBER”

11. Endif

1. end cancellation

7.2 CYCLOMATIC COMPLEXITY:

V(G) = Number of regions

=3

V(G)= Edges –nodes +2

= 13-12+2

=3

V(G)= Predicate nodes +1

= 2+1

=3

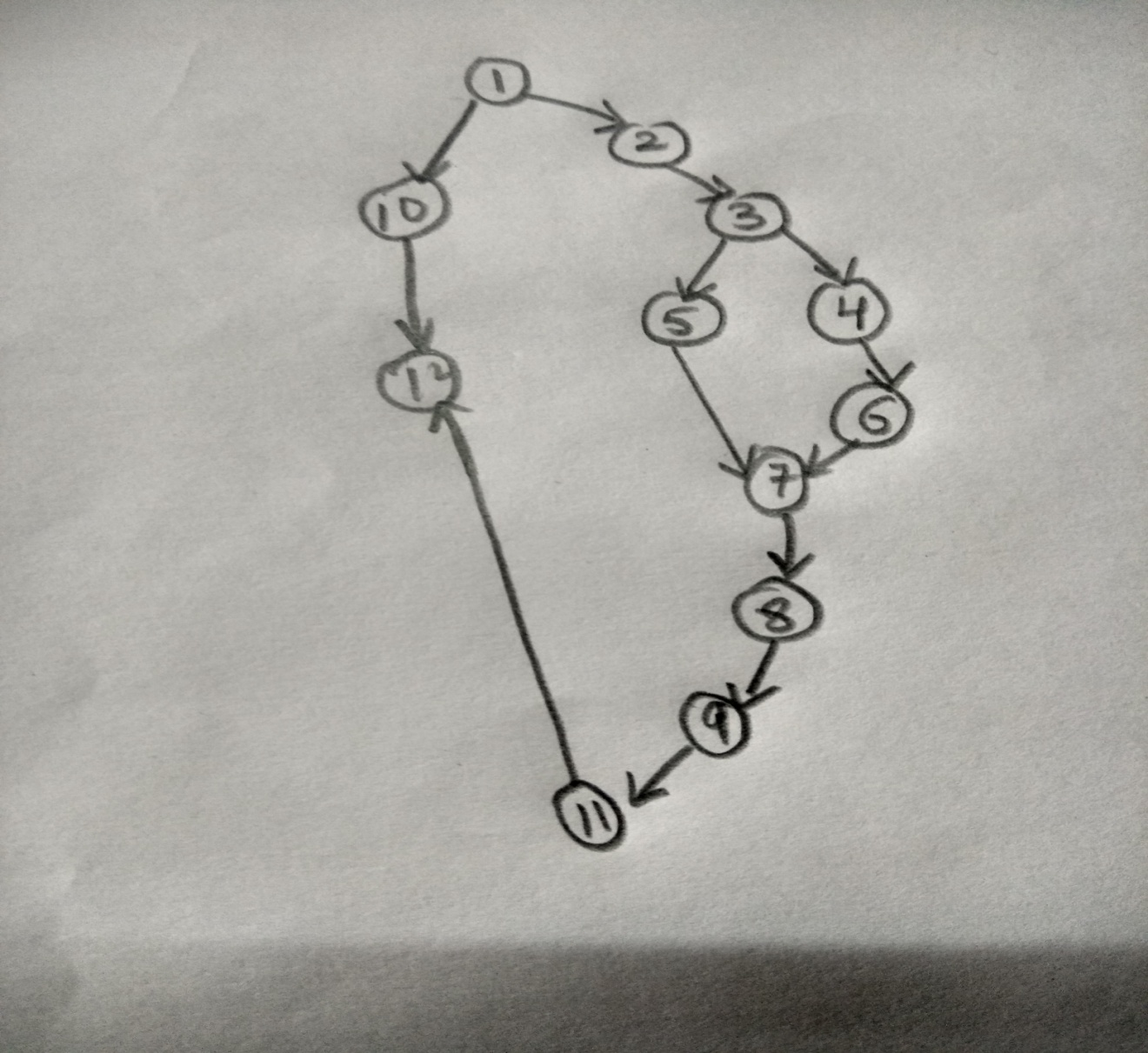
LINEARLY INDEPENDENT PATHS FOR FLOW GRAPH

Path 1 = 1-10-12

Path 2 = 1-2-3-4-6-7-8-9-11-12

Path 3 = 1-2-3-5-7-8-9-11-12

7.3CONTROL FLOW GRAPH



1. ANNEXURES







FORGET PASSWORD ?

LOGIN

ID

PASSWORD

GET YOUR TICKET

No. of External Inputs:4

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:1

No. of external interface files:0



NATIONALITY

ADDRESS

SECURITY ANSWER

SECURITY QUESTION

LAST NAME

OCCUPATION

PASSWORD

ID

FIRST NAME

DOB

REGISTER

No. of External Inputs:11

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:1

No. of external interface files:0



ENTER THE OTP

PHONE NUMBER

REGISTER

No. of External Inputs:3

No. of External Outputs:1

No. of External Inquiries:1

No. of internal logical files:1

No. of external interface files:0



ENTER THE OTP

EMAIL ID

REGISTER

No. of External Inputs:3

No. of External Outputs:1

No. of External Inquiries:1

No. of internal logical files:1

No. of external interface files:0



CANCELLATION

UPDATE TRAIN INFO

STATUS

HISTORY

BOOK SEAT

PAYMENT

SEARCH TRAIN

MENU

No. of External Inputs:7

No. of External Outputs:7

No. of External Inquiries:0

No. of internal logical files:1

No. of external interface files:0



DESTINATION

SOURCE

TATKAL OR GENERAL TICKET

SEARCH TRAIN

No. of External Inputs:4

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:1

No. of external interface files:0



DESTINATION STATION

SOURCE STATION

TRAIN NO.

DELIVERY TIME

PLATFORM NO.

DATE

ARRIVAL TIME

TRAIN NAME

TRAIN SCHEDULE

SEARCH TRAIN

No. of External Inputs:1

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:1

No. of external interface files:0



SELECT TRAIN NAME

SEARCH TRAIN

No. of External Inputs:2

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:0

No. of external interface files:0



VEG / NON-VEG / NO FOOD

CHAIR CAR OR SLEEPER

AC / NON AC

BOOK SEAT

No. of External Inputs:4

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:1

No. of external interface files:0



COACH NAME

SEAT NO.

PNR NO.

BOOK SEAT

No. of External Inputs:1

No. of External Outputs:4

No. of External Inquiries:0

No. of internal logical files:1

No. of external interface files:0



* MASTER CARD
* DEBIT CARD
* CREDIT CARD

MODE OF PAYMENT

PAYMENT

No. of External Inputs:2

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:0

No. of external interface files:0



CVV

VALID TILL

CARD NO.

PAYMENT

No. of External Inputs:4

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:0

No. of external interface files:1



PAYMENT SUCCESSFUL

PAYMENT



ENTER PNR NO.

CANCELLATION

No. of External Inputs:2

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:0

No. of external interface files:0



TRAIN NAME

REFUND AMOUNT

CANCELLATION NO.

CANCELLATION

No. of External Inputs:0

No. of External Outputs:3

No. of External Inquiries:0

No. of internal logical files:0

No. of external interface files:0



ENTER PNR NO.

CHECK STATUS

No. of External Inputs:2

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:0

No. of external interface files:0



STATUS

CHECK STATUS

No. of External Inputs:0

No. of External Outputs:0

No. of External Inquiries:0

No. of internal logical files:1

No. of external interface files:0



PAST TICKET RECEIPT

HISTORY

No. of External Inputs:0

No. of External Outputs:1

No. of External Inquiries:0

No. of internal logical files:1

No. of external interface files:0



DATE

ENTER TRAIN NAME

ENTER TRAIN NO.

ARRIVAL TIME

DESTINATION

SOURCE

DEPARTURE TIME

UPDATE TRAIN INFO.

No. of External Inputs:8

No. of External Outputs:0

No. of External Inquiries:0

No. of internal logical files:1

No. of external interface files:0