

Disperse Slot Intimation System for Streamlined Distribution in Civil Supplies Department

A Project Report Submitted to the Government Arts College (Autonomous), Coimbatore

In Fulfillment of the requirements For
the Award of Degree of

MASTER OF COMPUTER APPLICATIONS

Submitted by

JANANI T

REG. NO : 22MCA502

Under the Guidance of

Dr. R.A. ROSELINE M.Sc., M.Phil., Ph.D.,

Associate Professor and Head



POST GRADUATE AND RESEARCH DEPARTMENT OF COMPUTER APPLICATIONS

GOVERNMENT ARTS COLLEGE (AUTONOMOUS), COIMBATORE-641 018

Re-accredited with "B++" Grade by NACC. Affiliated to Bharathiyar University.

DECLARATION

I here declare that this project, entitled with “**DISPERSE SLOT INTIMATION SYSTEM FOR STREAMLINED DISTRIPTION IN CIVIL SUPPLIES DEPARTMENT**” Submitted to the Government Arts College (Autonomous), Coimbatore in fulfillment of the requirements for the award of the Degree of Master of Computer Applications is a bonafide record of the original project work done by under the Supervision and guidance of Dr. R.A. Roseline, M.Sc.,M.Phil., Ph .d., Associate Professor, Post Graduate and Research Department of Computer Applications , Government Arts College (Autonomous), Coimbatore- 641 018.

PLACE : Coimbatore

DATE :

Signature of the Candidate

(JANANI T)

22MCA502

CERTIFICATE

This is to certify that the project entitled "**DISPERSE SLOT INTIMATION SYSTEM FOR STREAMLINED DISTRIBUTION IN CIVIL SUPPLIES DEPARTMENT** " is a record of an original project work done by **JANANI T (22MCA502)** submitted to Bharathiyar University in fulfillment of the requirement for the award of the Degree of **MASTER OF COMPUTER APPLICATIONS**.

Head of the Department

Signature of the Guide

Submitted for the project viva-voce examination held on_____

Internal Examiner

External Examine

ACKNOWLEDGEMENT

I hereby take this opportunity to acknowledge my deep sense of gratitude and heartfelt thanks to our beloved Principal Dr. R. ULAGI M.Sc., M.Phil., Ph.D., Government Arts College (Autonomous), Coimbatore, who has given permission to carry out the project by providing necessary facilities to complete it successfully.

I also express my sincere and profound thanks to the Head of the Department, Dr. R. A. ROSELINE M.Sc., M. Phil., Ph.D., Associate Professor and Head, Post Graduate and Research Department of Computer Applications, Government Arts College (Autonomous), Coimbatore, for her valuable support and suggestions in my project work.

I would also express my sincere gratitude to my internal guide, Dr. R. A. ROSELINE M.Sc., M.Phil., Ph. D., Associate Professor and Head, Post Graduate and Research Department of Computer Applications, Government Arts College (Autonomous), Coimbatore, for who wrought shaped my project into better one with her valuable guidance for completion of this project.

Finally, I also express my sincere gratitude to all of my parents and friends who helped for my career without whose sustained support, I could not have made my debut in Computer Applications.

ABSTRACT

Public distribution system is a government-sponsored chain of shops entrusted with the work of distributing basic food and non-food commodities to the needy sections of the society at very cheap prices. Wheat, rice, kerosene, sugar, etc. are a few major commodities distributed by the public distribution system. Fair Price Shop does not open every day, nor do they keep regular hours. Even on the days that the Fair Price Shop is open, ration card holders have to stand in long queues. But due to delay in supply all citizen needs to come to the fair price shop and ask them whether they are providing the items today. As social distancing was not followed at several fair price shops during the first phase of public distribution, the Civil Supplies Department has issued paper token to the beneficiaries, mentioning the date for them to avail food grains and relief fund. The proposed project aims to modernize the Public Distribution System (PDS) in India, specifically addressing challenges faced by Fair Price Shops. By implementing a virtual queuing system through the Q-Learning algorithm, the approach seeks to replace traditional physical queues with organized slot allocations. Ration cardholders would receive SMS notifications specifying the date and time for product collection, reducing the need for individuals to stand in long queues or frequent the Fair Price Shop every day. This not only saves time for beneficiaries but also aligns with social distancing measures crucial for public health. The incorporation of Q-Learning, a reinforcement learning algorithm, showcases a forward-thinking approach to problem-solving. Additionally, the system allows for two re-slot allocations, providing flexibility for those who miss their initial collection slot. This not only streamlines the process but also allows individuals to view product details online, saving time and enhancing accessibility. This project not only addresses immediate challenges in the distribution process but also sets a precedent for leveraging technology to enhance efficiency and adaptability in government initiatives.

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
4.5	Database Design 4.5.1 Structure for table admin 4.5.2 Structure for table rq_consumer 4.5.3 Structure for table rq_employee 4.5.4 Structure for table rq_product 4.5.5 Structure for table rq_ration 4.5.6 Structure for table rq_stock 4.5.7 Structure for table rq_stock_req 4.5.8 Structure for table rq_timeslot 4.5.9 Structure for table rq_time_queue	28
6.2	Test cases	48

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1.1	PDS	1
1.2	Working of PDS	1
1.3	Fair Price Shop	2
1.4	Ration Card	3
1.5	Types of Ration Card	4
1.6	Working of RL	7
1.7	Taxonomy of RL Algorithms	8
1.8	Applications of RL	10
3.1	Deep Q-Learning	17
6.1	Boundary Testing	48

LIST OF APPREVATIONS

S.NO	APPREVATIONS	EXPLANATIONS
1	PDS	Public Distribution System
2	FCI	Food Corporation of India
3	TPDS	Targeted Public Distribution System
4	AAY	Antyodana Anna Yojana
5	PHH	Priority Household
6	BPL	Below Poverty Line
7	APL	Above Poverty Line
10	RL	Reinforcement Learning
11	MC	Monte Carlo
12	SARSA	State-action-reward-state-action
13	DQN	Deep Q neural network
14	PD	Priority Discipline
15	AVQ	Adaptive Virtual Queue
16	WSGI	Web Server Gateway Interface

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	DECLARATION CERTIFICATE ACKNOWLEDGEMENT ABSTRACT LIST OF TABLES LIST OF FIGURES LIST OF ABBREVIATIONS	
1	INTRODUCTION 1.1. Overview 1.2. Problem Statement 1.3. Artificial Intelligence 1.4. Objective of the project	1
2	SYSTEM ANALYSIS 2.1. Existing System 2.1.1. Disadvantages 2.2. Proposed System 2.2.2 Advantages	14
3	SYSTEM SPECIFICATION 3.1 Hardware Requirements 3.2 Software Requirements	19
4	SYSTEM DESIGN 4.1 Problem definition 4.2. Modules List 4.3 DFD diagrams 4.4 ER diagram 4.5 Database Design 4.6 Input Design	20

	4.7 Output Design 4.8 Concept Flow 4.9. System Architecture 4.10. System Flow	
5	SYSTEM DEVELOPMENT 5.1 Front end 5.2 Back end	39
6	SYSTEM TESTING & IMPLEMENTATION 6.1 Valid Payloads 6.2 Unit Testing 6.3 Data centric 6.4 Functionality Testing 6.5 Integration Testing 6.6 Boundary Testing 6.7 Test cases	44
7	CONCLUSION AND FUTURE ENHANCEMENT 7.1 Conclusion 7.2 Future Enhancement	50
8	APPENDICES 8.1 Source Code 8.2 Screenshot	51
9	REFERENCES	75
10	LIST OF PUBLICATIONS	

CHAPTER 1

INTRODUCTION

1.1. Overview

Public distribution system is a government-sponsored chain of shops entrusted with the work of distributing basic food and non-food commodities to the needy sections of the society at very cheap prices. Wheat, rice, kerosene, sugar, etc. are a few major commodities distributed by the public distribution system. The goal of the Public Distribution System in Tamil Nadu is to ensure food security to all citizens, particularly poor people, by making available essential commodities of good quality at affordable prices every month, through fair price shops which are easily accessible.



Figure 1.1. PDS

1.1.1. Public Distribution System

Public Distribution System (PDS) has evolved as a system of management of scarcity through distribution of food grains at affordable prices. Over the years, PDS has become an important part of Government's Policy for management of food economy in the country.

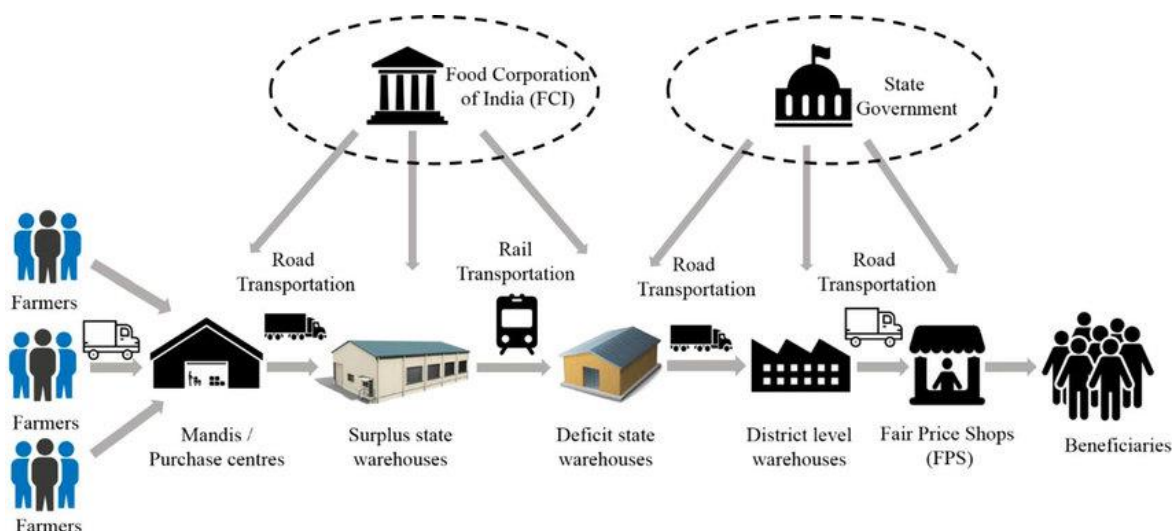


Figure 1.2. Working of PDS

It is additional in nature and is not intended to make available the entire requirement of any of the commodities distributed under it to a household or a section of the society. PDS is operated under the joint responsibility of the Central and the State Governments. The Central Government, through Food Corporation of India (FCI), has assumed the responsibility for procurement, storage, transportation and bulk allocation of food grains to the State Governments. The operational responsibility including allocation within State, identification of eligible families, issue of Ration Cards and supervision of the functioning of Fair Price Shops (FPSs) etc., rest with the State Governments. Under the PDS, presently the commodities like wheat, rice, sugar and kerosene are being allocated to the States /UTs for distribution. Some States/ UTs also distribute additional items of mass consumption through the PDS outlets such as pulses, edible oils, iodized salt, spices, etc.

1.1.2. Fair Price Shop or Ration Shop

Fair Price Shops are very common in India. We often find one fair price shop, commonly known as ration shops, in every locality of urban, semi-urban and rural areas.



Figure 1.3. Fair Price Shop

It has been licensed to distribute essential commodities by an order issued under section 3 of the Essential Commodities Act, 1955, to the ration card holders under the Targeted Public Distribution System. It is the responsibility of every State Government to establish institutionalised licensing arrangements for fair price shops in accordance with the relevant provisions of the Public Distribution System (Control) Order, 2001 made under the Essential Commodities Act, 1955, as amended from time to time for efficient operations of the Targeted Public Distribution System.

Under the Targeted Public Distribution System, it is the duty of the State Government to-

- take delivery of food grains from the designated depots of the Central Government in the State, at the prices specified,

- organise intra-State allocations for delivery of the allocated food grains through their authorised agencies at the door-step of each fair price shop; and
- ensure actual delivery or supply of the food grains to the entitled persons at the prices specified.

1.1.3. Ration Card

Ration card means a document issued under an order or authority of the State Government for the purchase of essential commodities from the fair price shops under the Public Distribution System (PDS) / Targeted Public Distribution System (TPDS).



Figure 1.4. Ration Card

Essentially it is a card / document that enables rationing of scarce commodities. In the present-day context, ration card may not be necessarily for the distribution of scarce commodities, rather, it is a general tool for implementing welfare measures. A Ration Card enables a citizen to procure essential commodities at a subsidised rate and also used to function as an important tool of identification (e.g., for obtaining Voter's ID / membership in electoral rolls etc. However, under Order, 2015.pdf TPDS (Control) Order, 2015 it is stipulated that Ration card shall not be used as a document of identity or proof of residence) Though ration cards have been in use for a very long time, the term got legally defined later in Section 2(16) of National Food Security Act, 2013. Ration Card is issued per family. It is voluntary and not compulsory for citizens to acquire it. However, all those citizens who want to get subsidised food may need to get that.

1.1.3.1. Types of Ration Card

Different types of ration cards are present in India. The state government categorises people and issues different ration cards according to the categories. In 2013, The National Food and

Security Act (NFSA) was passed to provide a certain quantity and quality food to people at affordable prices. NFSA provides for two types of ration cards for all the states in the country.



Figure 1.5. Types of Ration Card

- **Ration Cards Under NFSA, 2013**

NFSA provides for ration cards which are issued by the respective state governments. Distribution of food in fair price shops is according to the quantity and quality mentioned in NFSA. The different types of ration cards under NFSA are:

- **Antyodana Anna Yojana (AAY)**

This type of ration card is given to impoverished families identified by the state governments. Persons who do not have stable income are issued this card. Unemployed people, women and old aged people fall under this category. These card holders are eligible to receive 35kg of food grains per month per family. They receive food grains at the subsidised price of Rs.3 for rice, Rs.2 for wheat and Rs.1 for coarse grains.

- **Priority Household (PHH)**

The families not covered under AAY come under the PHH. The state governments identify priority household families under the Targeted Public Distribution System (TPDS) according to their exclusive, inclusive guidelines. The PHH cardholders receive 5kg of food grains per person per month. Food grains are at the subsidised price of Rs.3 for rice, Rs.2 for wheat and Rs.1 for coarse grains for these cardholders.

- **Ration Cards Under TPDS**

Before the introduction of NFSA, the state governments issued ration cards under the Targeted Public Distribution System (TPDS). After passing NFSA, states started issuing ration cards under it (which are mentioned above). The state governments which are yet to enforce the NFSA system, still follow the old ration cards issued by them under TPDS. They are:

- **Below Poverty Line (BPL)**

Families that have BPL cards are the ones who are living below the poverty line specified by the state government. BPL families receive 10kg to 20kg food grains per family per month at

50% of economic cost. The subsidised end retail price for specified quantities of wheat, rice, sugar and other items varies from state to state. Each state government fixes different rates per quantity.

- **Above Poverty Line (APL)**

Families that have this card are the ones who are living above the poverty line as specified by the state government. APL families receive 10kg to 20kg food grains per family per month at 100% of economic cost. Each state government fixes a subsidised retail rate for rice, wheat, sugar and kerosene oil for a certain quantity.

- **Annapoorna Yojana (AY)**

AY ration cards are given to older people who are poor and above 65 years. Cardholders receive 10 kgs of food grains per month under this card. State governments issue these cards to the older people who come under this scheme as specified by them.

The system is often blamed for its inefficiency and rural-urban bias. It has not been able to fulfil the objective for which it was formed. Moreover, it has frequently been criticized for instances of corruption and black marketing.

1.2. Problem Statement

Government ration shop is established to serve Indian citizen by providing the food materials for affordable cost. Ration Shop does not open every day, nor do they keep regular hours. Even on the days that the Ration Shop is open, ration card holders have to stand in long queues. But due to delay in supply all citizen needs to come to the ration shop and ask them whether they are providing the items today. Because of this problem when the items are provided people stand in big queue fighting for the items thinking that they won't get items. The people need to stand in a long queue to get the commodities which leads to waste of time for the people. This creates crowd in ration shops. It's strange that with all the advancement in technology, this mode has not changed much over the last millennium. If they are aware of the items content and when they are provided then it will be helpful for them. The model of waiting in lines is very inefficient, customers have to be in line in order to be served even if the line may have more than a hundred customers ahead. Despite this fact, service-based organizations that have the longest waiting time are making very little effort to improve on it. Needed is a system which allows queue management to be provided as a service and customers can accurately estimate their wait times and keep them updated as the queue progresses without having to wait in the premise or crowded lobbies. Physically queueing is a reality on many industries that provide services or sell goods. Queueing problems exist when multiple people need access to a resource, and the service cannot match the level of demand. On the other hand, queues may get

unnecessary big even for this purpose and may lead to long idle times for the customers. Waiting in a queue can be stressful and exhausting for the clients because of the enforced idle time, and may lead to decreased customer satisfaction. Queueing lines can also be observed in everyday life activities, such as paying at groceries, waiting for a table at a restaurant, or waiting to order at a fast-food restaurant and also at ration shop. Automated queue management system is widely used in the industry, including banking. The scheduling algorithm needs automatically optimize resource utilization based on the changing demand. However, most of these queue management systems do not have waiting-time estimation features. To the best of our knowledge, there is currently no generic queueing system, where authorised person can set up a queue that anyone can nonsqueezing optimization techniques are used in several industries to improve customer service. So, the algorithm has to schedule the cards at once without any prior experience and prepared information. Our system is not only intimate the citizens about their sales, but also provides schedule for buying the food items. A virtual queuing is one of the most modernized and technologically advanced solution to deal with the wait time and other queue management problems.

1.3. Artificial Intelligence

At its simplest form, artificial intelligence is a field, which combines computer science and robust datasets, to enable problem-solving. It also encompasses sub-fields of machine learning and deep learning, which are frequently mentioned in conjunction with artificial intelligence. These disciplines are comprised of AI algorithms which seek to create expert systems which make predictions or classifications based on input data.

1.3.1 Artificial Intelligence History

The idea of 'a machine that thinks' dates back to ancient Greece. But since the advent of electronic computing (and relative to some of the topics discussed in this article) important events and milestones in the evolution of artificial intelligence include the following:

- 1950: Alan Turing publishes *Computing Machinery and Intelligence*. Turing—famous for breaking the Nazi's ENIGMA code during WWII—proposes to answer the question 'can machines think' and introduces the Turing Test to determine if a computer can demonstrate the same intelligence (or the results of the same intelligence) as a human. The value of the Turing test has been debated ever since.
- 1956: John McCarthy coins the term 'artificial intelligence' at the first-ever AI conference at Dartmouth College. (McCarthy would go on to invent the Lisp language.)

- 1967: Frank Rosenblatt builds the Mark 1 Perceptron, the first computer based on a neural network that 'learned' through trial and error. Just a year later, Marvin Minsky and Seymour Paper publish a book titled Perceptron's, which becomes both the landmark work on neural networks and, at least for a while, an argument against future neural network research projects.
- 1980s: Neural networks which use a backpropagation algorithm to train itself become widely used in AI applications.
- 1997: IBM's Deep Blue beats then world chess champion Garry Kasparov, in a chess match (and rematch).
- 2011: IBM Watson beats champions Ken Jennings and Brad Rutter at Jeopardy!
- 2015: Baidu's Minwa supercomputer uses a special kind of deep neural network called a convolutional neural network to identify and categorize images with a higher rate of accuracy than the average human.
- 2016: DeepMind's AlphaGo program, powered by a deep neural network, beats Lee Sodol, the world champion Go player, in a five-game match. The victory is significant given the huge number of possible moves as the game progresses (over 14.5 trillion after just four moves!). Later, Google purchased DeepMind for a reported \$400 million.

1.3.2. Reinforcement Learning

Reinforcement learning deals with artificial intelligence that allows machines to work independently without any manual interpretation to reach their goals and to interact with dynamic environment. In Reinforcement Learning (RL), agents are trained on a reward and punishment mechanism. The agent is rewarded for correct moves and punished for the wrong ones. In doing so, the agent tries to minimize wrong moves and maximize the right ones.

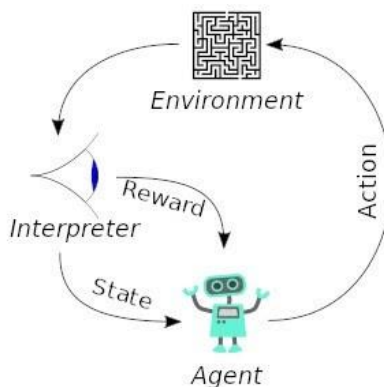


Figure 1.6. Working of RL

1.3.2.1. Approaches for implementing reinforcement learning

There are generally three ways to implement a reinforcement learning algorithm: value-based, policy-based, or model-based. These approaches determine how the agent will take action and interact with the environment.

- **Value-based**

This approach is about finding the optimal value function, which is essentially the maximum value at a state under any policy.

- **Policy-based**

In this approach, the agent tries to develop a policy so that the action performed in every state would help maximize the future reward.

- **Model-based**

In this approach, a virtual model is created for each environment, and the agent explores it to learn. Since the model representation is different for each environment, there isn't a particular RL algorithm or solution for this approach.

1.3.2.2. Taxonomy of Reinforcement Learning

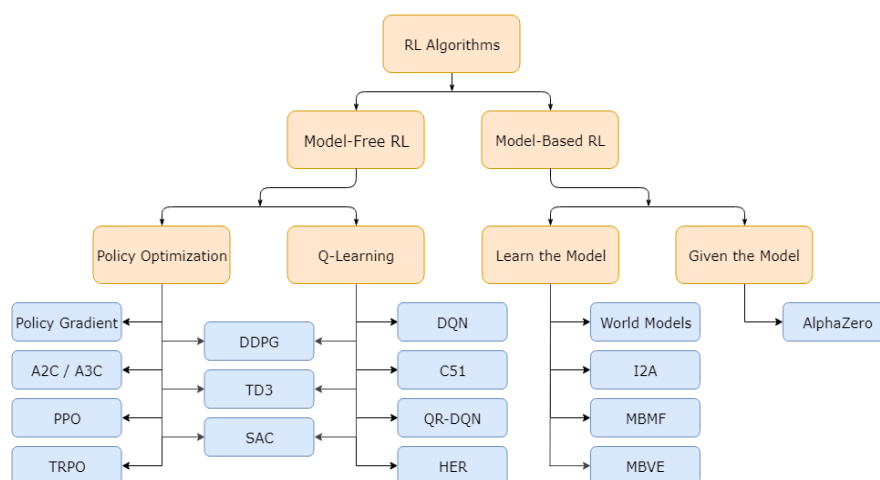


Figure 1.7. Taxonomy of RL Algorithms

Algorithms which use a model are called model-based methods, and those that don't are called model-free. While model-free methods forego the potential gains in sample efficiency from using a model, they tend to be easier to implement and tune. model-free methods are more popular and have been more extensively developed and tested than model-based methods.

1.3.2.3. Reinforcement learning algorithms

Reinforcement learning algorithms can be classified into two: model-free RL algorithms and model-based RL algorithms. Q-learning and deep Q learning are examples of model-free RL algorithms.

- **Q-learning**

Q-learning is a value-based RL method of providing information. It's used for temporal difference learning and determines how good an action is at a particular state. Q-learning is an off-policy learner, meaning the agent will learn the value function based on the action derived from another policy. Q-learning starts with the initialization of the Q-table. Then the agent selects an action and performs it. The reward for the action is measured, and then the Q-table is updated. A Q-table is a table or matrix created during Q-learning. After each action, the table is updated. In Q-learning, the agent's goal is to maximize the value of Q. In this method, the agent strives to find the best action to take at a particular state. The Q stands for quality, which indicates the quality of action taken by the agent.

- **Monte Carlo Method**

The Monte Carlo (MC) method is one of the best ways an agent can get the best policy to gain the highest cumulative reward. This method can be used only in episodic tasks, which are tasks that have a definite end. In the MC method, the agent learns directly from episodes of experience. This also means that the agent initially has no clue about which action leads to the highest reward, so the actions are chosen randomly. After selecting a bunch of random policies, the agent will become aware of the policies that lead to the highest rewards and get better at picking policies.

- **SARSA**

State-action-reward-state-action (SARSA) is an on-policy temporal difference learning method. This means that it learns the value function based on the current action derived from the currently used policy.

SARSA reflects the fact that the main function used to update the Q-value depends on the agent's current state (S), the action chosen (A), the reward it gets for the action (R), the state the agent enters after performing the action (S), and the action it performs in the new state (A).

- **Deep Q neural network**

Deep Q neural network (DQN) is Q-learning with the help of neural networks. It's ideal when the state and action spaces are significant, as defining a Q-table will be a complex and time-consuming task. Instead of a Q-table, neural networks determine the Q-values for each action based on the state.

1.3.2.4. Application of Reinforcement Learning

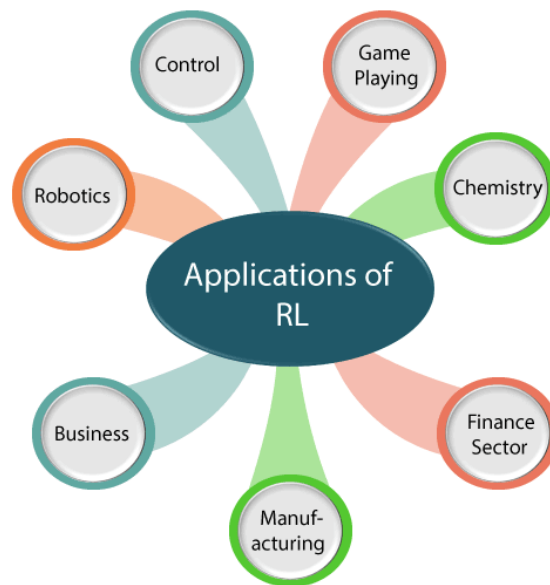


Figure 1.8. Applications of RL

- **Traffic Light Control**

In the article "multi-agent system based on reinforcement learning to control network traffic signals," the researchers tried to design a traffic light controller to solve the congestion problem. Tested only in a simulated environment, their methods showed results superior to traditional methods and shed light on multi-agent R.L.'s possible uses in traffic systems design. Five agents were placed in the five intersections traffic network, with an R.L. agent at the central intersection to control traffic signalling. The state was defined as an eight-dimensional vector, with each element representing the relative traffic flow of each lane. Eight options were available to the agent, each representing a combination of phases, and the reward function was defined as a reduction in delay compared to the previous step. The authors used DQN to learn the Q value of {state, action} pairs.

- **Auctions and Advertising**

Researchers at Alibaba Group published the article "Real-time auctions with multi-agent reinforcement learning in display advertising." They stated that their cluster-based distributed multi-agent solution (DCMAB) has achieved promising results and, therefore, plans to test the Taobao platform's life.

Generally speaking, the Taobao ad platform is a place for marketers to bid to show ads to customers. This can be a problem for many agents because traders bid against each other, and their actions are interrelated. In the article, merchants and customers were grouped into different groups to reduce computational complexity. The agents' state-space indicated the

agents' cost-revenue status, the action space was the (continuous) bid, and the reward was the customer cluster's revenue.

- **Web Systems Configuration**

There are more than 100 configurable parameters in a Web System, and the process of adjusting the parameters requires a qualified operator and several tracking and error tests.

The article "A learning approach by reinforcing the self-configuration of the online Web system" showed the first attempt in the domain on how to autonomously reconfigure parameters in multi-layered web systems in dynamic VM-based environments.

The reconfiguration process can be formulated as a finite MDP. The state-space was the system configuration; the action space was {increase, decrease, maintain} for each parameter. Finally, the reward was defined as the difference between the intended and measured response times.

- **Industrial automation**

Thanks to the reinforcement learning capabilities from DeepMind, Google was able to reduce energy consumption in its data centres dramatically. Bonsai, recently acquired by Microsoft, offers a reinforcement learning solution to automate and “build intelligence into complex and dynamic systems” in energy, HVAC, manufacturing, automotive and supply chains.

- **Enhance predictive maintenance**

Machine learning has been used in manufacturing for some time, but reinforcement learning would make predictive maintenance even better than it is today.

- **Advanced algorithms**

Being developed and combined in new ways to analyse more data faster and at multiple levels. This intelligent processing is key to identifying and predicting rare events, understanding complex systems and optimizing unique scenarios.

- **APIs, or application programming interfaces**

That are portable packages of code that make it possible to add AI functionality to existing products and software packages. They can add image recognition capabilities to home security systems and Q&A capabilities that describe data, create captions and headlines, or call out interesting patterns and insights in data.

- **Customer Service**

Online chatbots are replacing human agents along the customer journey. They answer frequently asked questions (FAQs) around topics, like shipping, or provide personalized advice, cross-selling products or suggesting sizes for users, changing the way we think about customer engagement across websites and social media platforms. Examples include

messaging bots on e-commerce sites with virtual agents, messaging apps, such as Slack and Facebook Messenger, and tasks usually done by virtual assistants and voice assistants.

- **Automated stock trading**

Designed to optimize stock portfolios, AI-driven high-frequency trading platforms make thousands or even millions of trades per day without human intervention.

1.4 Objective of the Project

The system can be used by large number of organizations, shops, retail service providers having a huge number of customers to manage on daily basis. Any registered user can join a queue by using the virtual id of respective organization provided the organization or shop or retail service provider is using the same system to manage queues. Our system is flexible for user interaction more easily to communicate with admin. This technique can able to reduce the corruption related with ration system. The system would also provide access to an administrator dashboard with key metrics for showing various reports and statistics on status of queues and also permissions that would enable authorized users to update, manage, and view queues. Administrative dashboards would also contain daily queue statistics, call centres, Department data, counter and staff data and Reports view. The mobile app was to provide a list of service centres and booking options. This System can be freely used by any type of industry, groups etc. The System is highly scalable and reliable to be implemented for small to large number of family cards. Ration card Holders can Avoid standing in rows for a long time and there is no situation that Ration is completed in the Ration Shop. Users will have Complaint forum to post their complaints regarding Ration Shop and Admin will check their Complaints and Ration Card Holders can check the prices of the Groceries and Stock availability in their dashboard. The key purpose of the system is to develop a Slot Prediction based Virtual Queue System using Deep Q Learning that addresses the abovementioned problems and allow queue management to be provided as a service thereby minimizing waiting lines in service-oriented environments. The main purpose of this project is to develop and provide a web based generalized Virtual Queue and Slot Notification for Ration Distribution System. Then, the customer gets to know the slot time and date of the ration shop through the alert message and then he/she will go and get the required product at the reserved time. Citizen only visit the location once they received the SMS. This eliminates the need for crowded and helps practice social distancing.

The project was to achieve the following objectives:

- To analyse how existing system implement service queues.

- To develop a system that allows the use of virtual queues in web-based and mobile platforms.
- To implement USSD and messaging services for notifications.
- To provide security of the information gathered by the system.
- Every shop's geolocation will be located on the map.
- All the information regarding shop and shopkeeper respective to the ration card and region.
- Shop opening and closing details will be available.
- Stock provided based on category.
- Retailers can add, edit, delete and update their shop's other items also.
- Retailers can advertise their shops also.
- The Stock availability will be notified on regular basis.
- Product delivery feature.
- Complaint forum for complaints by users.
- The Admin and Moderator features for government and shopkeepers.
- Authentication system by Aadhaar and OTP.

CHAPTER 2

SYSTEM ANALYSIS

2.1. Existing System

The Civil Supplies Department has issued coupons to the beneficiaries, mentioning the date for them to avail food grains, commodities and relief fund. Here are some existing virtual queue management systems are

- Virtual Sign-up
- Online Channels
- Mobile Apps
- QR Code
- Traditional Channels
- Virtual Waiting/Queues
- Multi-channel Remote Queuing Information Updates
- Traditional Queue Management Components
- Counter Plates
- Digital Signage Screens
- Ticket Dispensing Kiosk/Terminal (Optional)
- Audio-Visual Announcements and Customer Calling
- Administrative Control Panels and Agent Dashboard
- Server Application or Queue Management Server
- Third-party System Integration
- Customer Feedback Module/System

There are many already providing services for digital queue management using various techniques such as

- **Kiosk Based Systems**

A kiosk is a small, stand-alone booth typically placed in high-traffic areas for business purposes. In this systems user can enter their details and receive the token either in the form of paper based or digital acknowledgement. However, these systems require users to physically visit the location in order to use the service.

- **Standard Linear Queues**

This form of queuing represents the normal or standard queue system where each service desk or cashier has a separate line.

- **Single Line Queues**

Also referred to as a Call Forward System, a single line groups customers and then feeds them to multiple cashiers or service areas. Often, Individual service stations are allocated to different service personnel part of one long counter or desk.

- **Dispersed or “Digital” Queues**

This type of queue management disperses waiting lines by offering a ticketing management system.

- **First Come, First Served (FCFS)**

This mode is commonly applied real-world situations, such as tellers in a bank.

- **Last Come, First Served (LCFS)**

This mode acts a reverse order service given to customer against their arrival.

- **Priority Discipline (PD)**

Under this discipline, customers are classified into categories, then each category is given different priorities.

- **Round Robin Scheduling**

Round robin scheduling (RRS) is a job-scheduling algorithm that is considered to be very fair, as it uses time slices that are assigned to each process in the queue or line. Each process is then allowed to use the CPU for a given amount of time, and if it does not finish within the allotted time, it is pre-empted and then moved at the back of the line so that the next process in line is able to use the CPU for the same amount of time.

- **Adaptive Virtual Queue (AVQ)**

The motivation behind the AVQ algorithm is to design an AQM scheme that results in a low-loss, low-delay and high utilization operation at the link. The AVQ algorithm maintains a virtual queue whose capacity (called virtual capacity) is less than the actual capacity of the link. An important feature of the AVQ algorithm is that one can employ any AQM algorithm in the virtual queue.

2.1.1 Disadvantages

- Existing techniques are full of complications and queues tend to have long waiting hours.

- The model of waiting in lines is very inefficient, customers have to be in line in order to be served even if the line may have more than a hundred customers ahead.
- Regional Disparities
- Urban Bias
- PDS dealers are sometimes found resorting to malpractices like diverting the grains to open market to get better margin
- The irregular opening of the shops
- The drawbacks i.e., manual process, time consumption, stock corruption etc.

2.2. Proposed System

The proposed system is basically divided into two subsystems the web server and the distribution system. The proposed system uses real time tokens over the web servers for managing the queues. The users can conveniently use the system with a web or android based application which allows a card holder to virtually join a queue. The whole system is centralized and hence users can have freedom to access the application from any device at any location. This system can be used without the need of physical availability of card holder for issuing the token, the FPS admin can generate tokens using just with web or android application as per his own convenience. The service providers or organizations can manage the queuers in real-time using the web server and queuers can get real-time updates regarding current status of queue using personalized notification.

Proposed an RL-based framework that takes a QoS constraint as input, and provides a dynamic service-rate control algorithm that satisfies the constraint without overuse of the service resources. This makes our method distinct from the existing service-rate control algorithms that quantify their performance by the achieved overall reward, which is highly dependent on the reward definition and might have no practical interpretations. Proposed a Deep Q Learning Model within the queueing framework to investigate equilibrium strategies in terms of capacity and number of Slots.

Deep Q-Learning Algorithm

As one of the main reinforcement learning algorithms, Q learning is a model-free learning method which provides the intelligent system with the ability to select the optimal action according to the action sequences from experience in the Markov environment. A key assumption of Q-learning is that the interaction between the agents and the environment can be treated as a Markov decision process (MDP), i.e., the current state and action of the agent

will determine the state transfer probability distribution and the next state with an immediate reward.

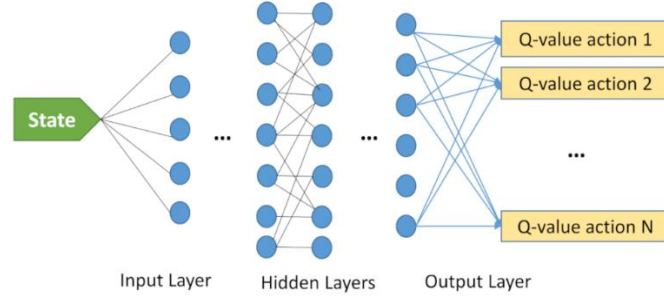


Figure 3.1. Deep Q-Learning

The goal of Q-learning is to find a policy that can maximize the reward. The Q-value is an important parameter in Q-learning. It is defined as the sum of rewards for executing the current related actions and those to be performed subsequently in accordance with a certain strategy. A given state s and action a correspond to a given Qvalue $Q(s,a)$. Q-value is used in the learning process to select the action. If the subsequent actions are performed according to the optimal polices the corresponding Q-value is referred to as the optimal Q value Q^* ,

$$Q^*(s,a) = r(s,a) + \gamma \sum T(s,a,s') \max_{a'} Q^*(s',a'),$$

where $T(s,a,s')$ represents the transfer probability from state s to state s' via action a , $r(s,a)$ represents the reward for executing action a from state s , $\gamma \in (0,1)$ is the discount factor, which indicates the degree of farsightedness. If the γ value is small, the system pays attention to only the recent actions. If γ is large the actions during a relatively long period of time are involved. An agent learning process can be viewed as selecting an action from a random state using a strategy. The value of $Q(s,a)$ is updated according to

$$Q_{t+1}(s,a) = (1 - \alpha) + Q_t(s,a) + \alpha [\gamma \max_{a'} Q(s',a')]$$

where $\alpha \in (0,1)$ is the learning factor used to control the speed of learning: the greater the value of α , the faster the convergence speed. After performing the selected action, the agent observes the new state and the reward obtained, and then updates the Q-value of the state and action based on the maximum Q-value of the new state. In this way the agent continually updates the action according to the new state until it arrives at the terminal state with an optimal Q-value Q^* .

Time Slot Length (T)

The length of the time slot is another design parameter that can affect the performance of the controller and the learned policy. In general, decreasing the time slot length provides finer-grained control over the system and can result in better optimized policies. However, choosing very small time slot lengths can cause two major practical problems. First, any controller has a limited speed due to its processing time and therefore, might not be able to interact with the environment in arbitrary short time-scales. The second and more important problem is that queueing systems do not respond to the actions instantaneously, which makes the policy learning problem even more complex. As a result, the time slot length should be large enough such that the immediate reward, provides a good assessment of the taken action. On the other hand, choosing very large time slot lengths can result in various issues too. As discussed earlier, larger time slot means less frequent control over the system and as a result, potentially less optimal control policies. Furthermore, if the time slot length becomes large enough such that the queueing system stabilizes after taking each action, the rewards become less dependent on the states and only assess the taken action (chosen service rates). Therefore, the learned actions become almost independent of the states, which questions the whole point of using adaptive service rate control.

- Provide customers with their slot time and date by SMS
- Card holders can able to set their preferred time by using this dashboard

2.2.1 Advantages

- Reduce customer walkaways by 60%
- Centrally Controllable & Easy to Use
- Organizes Service Area and reduces customer waiting times
- Enables efficient management of customer journey
- Optimizes Staff Performance & increase retention
- No Lines Smarter Citizens and Enhance customer communication and engagement
- Seamlessly manage customer flow.
- Monitor your workflows in real-time.
- Enabled Safer Socially Distanced Interactions
- Proper and well planning of Slot information.
- Social Distancing and Corruption is avoided by this proposed system.
- The customer feels more confident as they know no one else can take their turn and everyone will be served as per their turn.
- The automatic management eliminates all human related errors and problems.

CHAPTER 3

SYSTEM SPECIFICATION

3.1. Hardware specification

- Processors: Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 8 GB of DRAM
- Disk space: 320 GB
- Operating systems: Windows® 10, macOS*, and Linux*

3.2. Software specification

- Server Side : Python 3.7.4(64-bit) or (32-bit)
- Client Side : HTML, CSS, Bootstrap
- IDE : Flask 1.1.1
- Back end : MySQL 5.
- Server : WampServer 2i
- DL DLL : TensorFlow, Pandas, SiKit Learn

CHAPTER 4

SYSTEM DESIGN

4.1. Problem Definition

In this project the Ration Shop does not open every day, nor do they keep regular hours. Even on the days that the Ration Shop is open, ration card holders have to stand in long queues. But due to delay in supply all citizen needs to come to the ration shop and ask them whether they are providing the items today. Because of this problem when the items are provided people stand in big queue fighting for the items thinking that they won't get items. So the Social Distancing and Corruption is avoided by this proposed system. Evaluations show that the algorithm decreases queuing delay for dynamic traffic with more planning. This system will help to avoid the corruption in rationing system to a large extent by providing transparency at each level. This proposed project is for the benefit of common people and the government by avoiding crowd in public areas while product distribution. Also, when implemented in the Ration shops across Tamil Nadu, the government official can have a check on the on-going of each and every transaction done in the ration shops from their head office and this access of information privilege is given to every citizen of India as a Right to Information under the Act of Consumer Rights.

4.2. Modules List

1. QLess Ration Distribution Web App
2. Access Control Module
3. Time Slot Generator
4. End User Module
 - 4.1. Super Admin
 - 4.2. Shop Admin
 - 4.3. Citizen
5. Auto Notification Engine
6. Reports
7. Performance Analysis

4.2.1 Modules Description

4.2.1.1 QLess Public Distribution Web Application

This is the main module, which is used to design website and which would provide the following facilities.

4.2.1.2 Time Slot Generator

In this module, the time slot generator generates the time slot for each and every consumer with token ID. E-Token will generate a token ID that helps to keep track of the queue from anywhere in the world using internet. In E-Token We have a feature such as provide a token, view a token status, update a token, cancel a token, delete a token and many more. With these features we can easily manage the online queue system in any suitable domain without any hassle. The tokens will have a specific day and time when the ration cardholders can collect their share. Consumers notify by SMS and email they can collect them from their ration shops on the specified day and time, including the extra rice and relief fund announced as part of COVID-19 relief measures.

It uses the following parameters to generate time slot for card holders.

- Start Date/End Date: Select the first and last date for your time slots with the same criteria being entered.
 - Start Time/End Time: This would be the start time of the first game and end time of the last game if the time slots are being added back-to-back.
 - Weekdays: Select the days of the week the time slots will be added to.
- Conflict Handling: Choose one of the following:
 - Abort (Cancel the operation and not add/edit any slots)
 - Skip (Leave conflicting slots unchanged)
 - Update (Update/Overwrite conflicting slots to match)
- Slot Duration: Choose how long each time slots should be.
- Slot Type: allotted, Preferred Time and unallotted

Once all of the information has been updated click Save Changes and your time slot will appear on the website for each consumer separately. Time Slots is a feature that must be activated by ration shop admin.

4.2.1.3 Web Admin

Step 1: Register

Step 2: Login

Step 3: Add or Update or Delete

- **Ration Shop Admin Master:** This module is used to add the area wise ration shop staff and their details. This module supports the administrator to manage the ration centre details and administrator having rights to create ration centre account and administrator can also block and unblock the account.
- **Item Management:** This module supports the administrator to create items by providing item name, description & rate these items will be displayed for ration centre & consumer.
- **Area Master:** This module is used to add the area name.
- **Area Ration card details:** It is used to add the details of the head and members of the family. In this module administrator can also create the consumer account and administrator will have rights to block and unblock the consumer account and consumer family members.
- **Manage Supplier:** In this module supports administrator to manage the item supplier details of the company and it supports to manage the contact person of the manufacturing company.
- **Manage Forwarded:** The forwarded one who deliver the items to administrator to ration centre & ration centre to consumer this module supports to manage the detail of the item transportation.
- **Item Request:** This module supports to maintain the item request details raised by the ration centre and it supports to transfer the items from administrator to ration centre if the stock is available.
- **Stock Master:** Add stock, allot stock area wise and notify the stock availability. This module supports the administrator to manage stock details of the item and we have provided option to import the product by providing item details, quantity, supplier & forwarded details.

Step 4: Report Maintenance

4.2.1.4 Ration Shop Admin

In this module the ration shop admin can enter with their login credentials. The shop admin has a live dashboard to generate time slot and token id. The shop admin also keeps track of queuing numbers, wait times and service times. Schedule, reschedule and cancel appointments digitally to manage customer traffic and reduce operational costs. The shop admin can have unique registration id to maintain their details in database. Admin logged

in to their page, they will verify the stocks are available or anything to order from the government and maintain the product, employee and customer records.

4.2.1.5 Consumer

In this module, consumer can login into his account to see the collected ration details. consumer can also request to make changes in his account details to distributor through his login. Customer can view the available products in their home page. This module consists of the details of the customer who gets the notification of date and time to buy the stock from the ration shop. The consumer can then set the possible time options and also request for reschedule the date and time. They can able to view the purchase history by simply logged with their card number.

- **Purchase details:** This module provides the history of the item purchased by the Consumer.

4.2.1.6 Automatic Notification System

This module generates the OTP for consumer login these OTP send through by SMS and Email of the consumer. Another SMS and Email with the Date and Time to buy the ration commodities sent to the consumer mobile number and mail id provided.

Auto generated SMS are

- OTP
- Time slot
- Reschedule time slot
- Relief fund
- Thank you

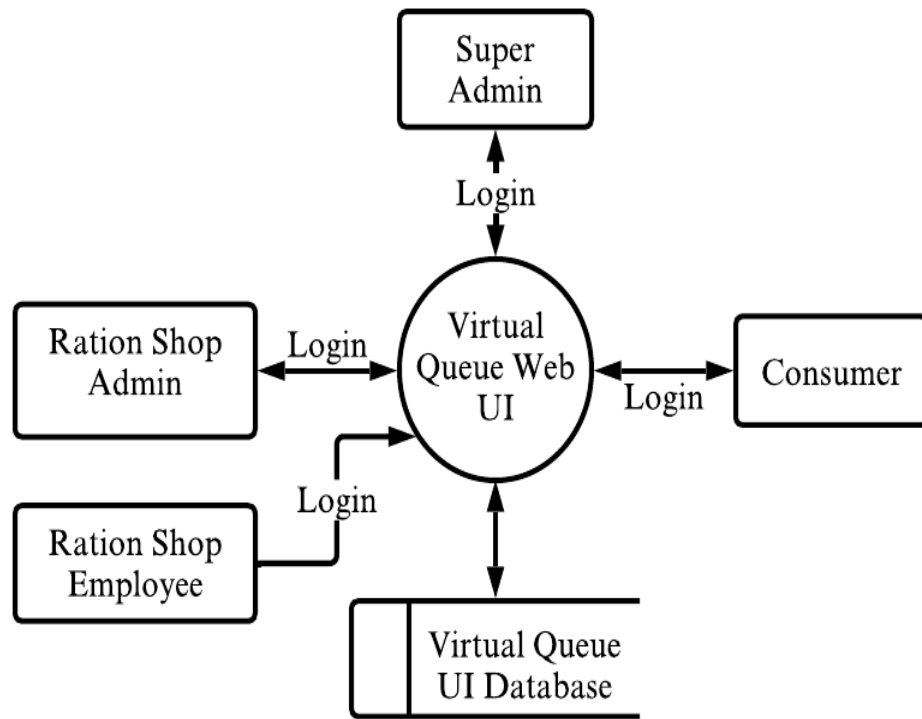
4.2.1.7 Performance Analysis

The proposed DQL algorithm was validated based on the variety of conditions and parameters. First, we calculated the performance indicator of each test instance under the action of the DQN algorithm with different delay factors. The following performance characteristics:

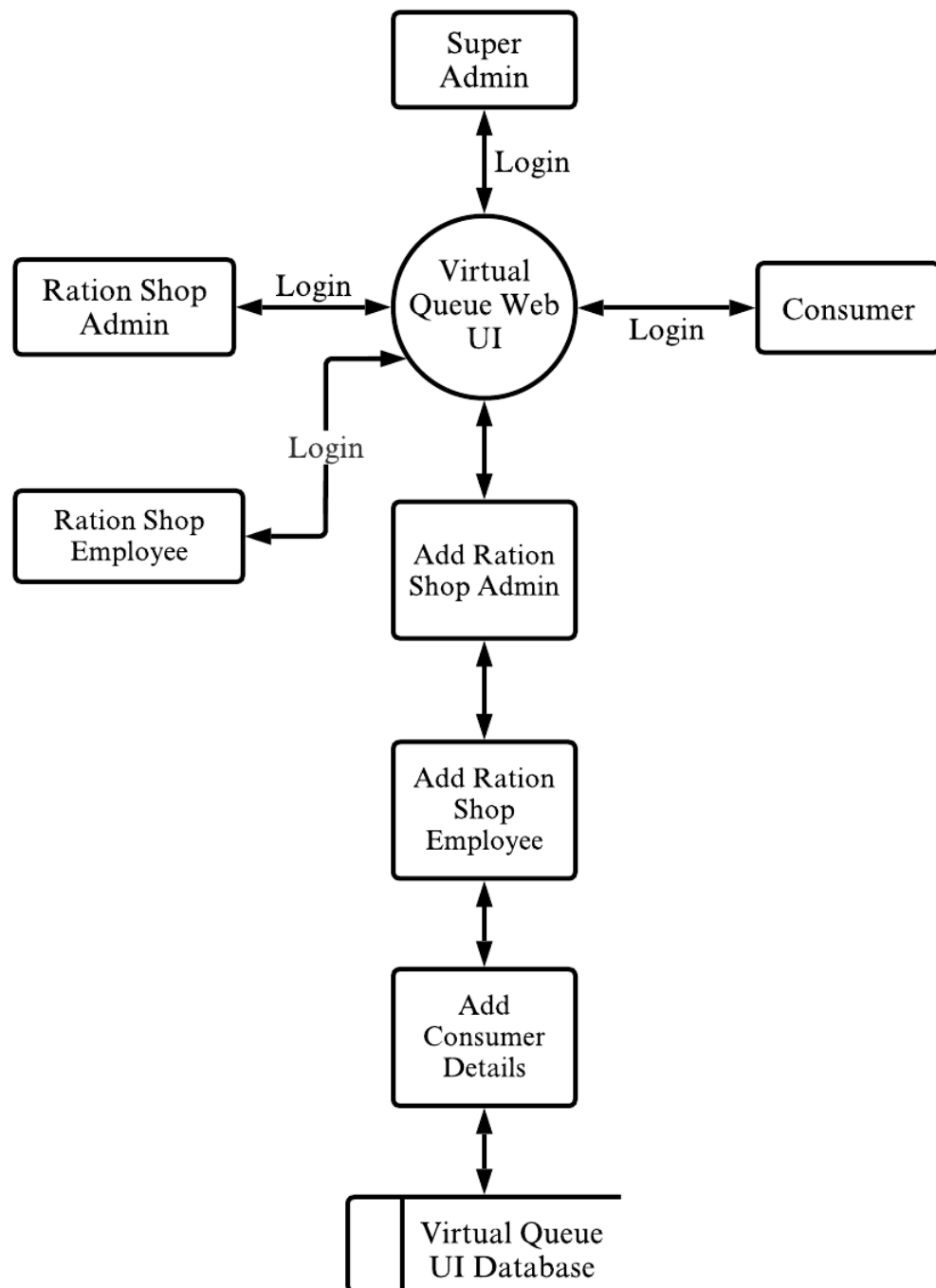
- Average queue length;
- Average time spent for every slot
- Queue Delay
- Throughput
- Slot utilization
- Time Interval

4.3 DFD diagrams

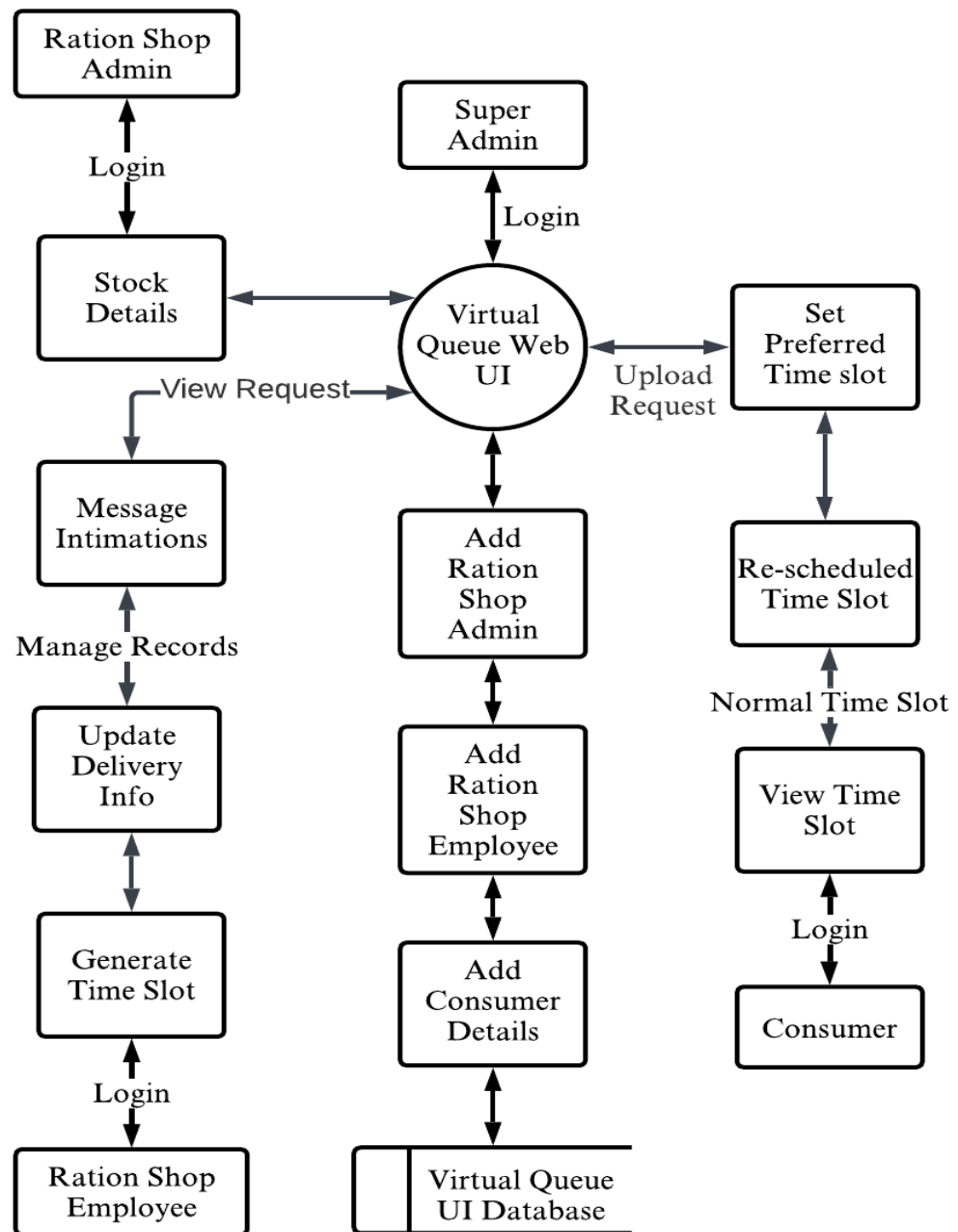
Level 0



DFD Level 0

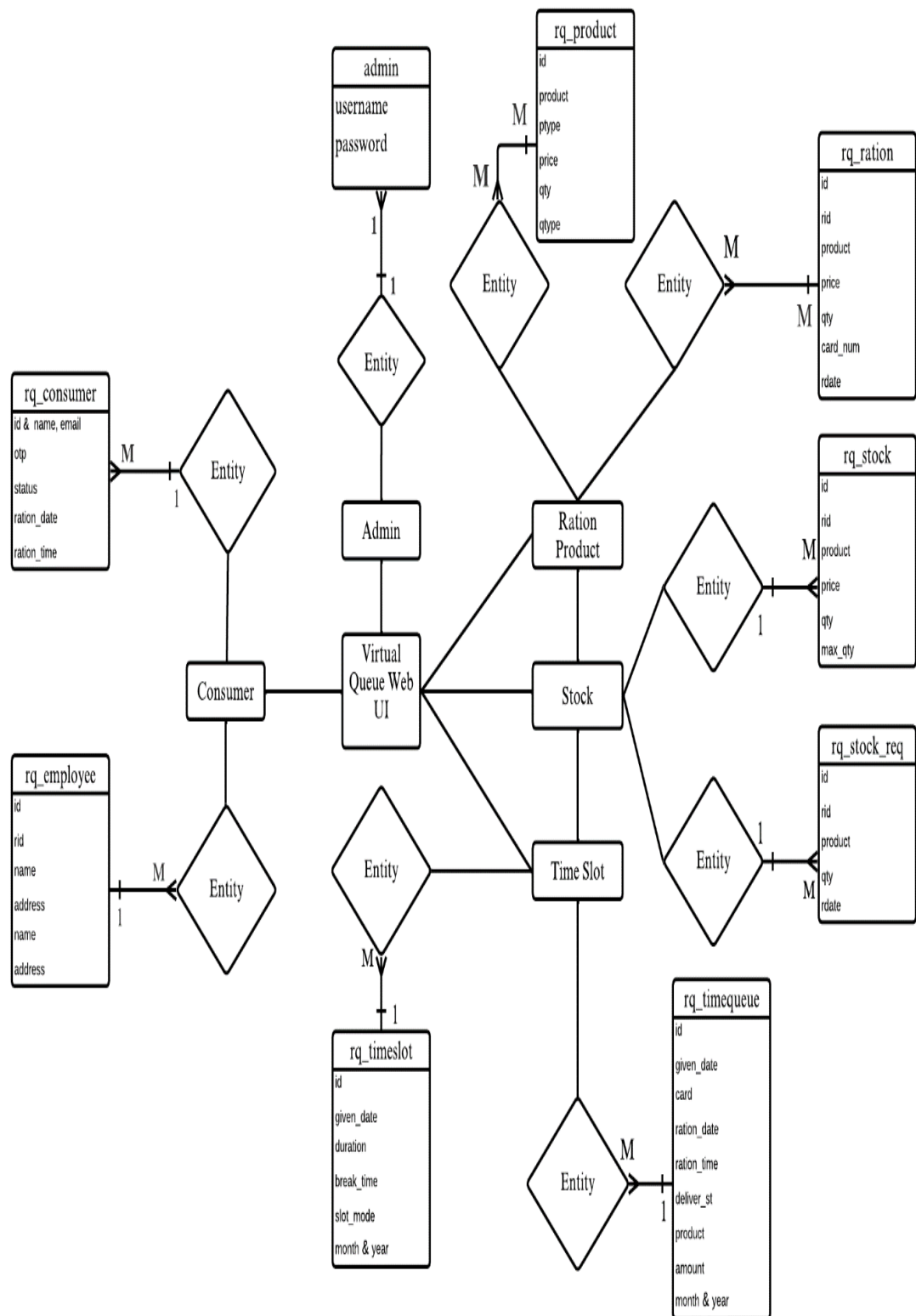
Level 1**DFD Level 1**

Level 2



DFD Level 2

4.4 ER diagram



ER Diagram

4.5 Database Design

Table 4.5.1 structure for table admin

Field	Type	Constraints	Descriptions
username	varchar(20)	Unique key	Admin has create username for unique key
password	varchar(20)	NULL	Admin has a create separate password

Table 4.5.2 structure for table rq_consumer

Field	Type	Constraints	Descriptions
id	int(11)	Primary key	Admin are create the consumer id
rid	varchar(20)	NOT NULL	This id admin are created
name	varchar(20)	NOT NULL	Enter the name
<i>card</i>	varchar(20)	NOT NULL	Enter the card details
mobile	bigint(20)	NOT NULL	Enter the contact numbers
email	varchar(40)	NOT NULL	Enter a email id
address	varchar(50)	NOT NULL	Enter the address
prefer_day	varchar(100)	NOT NULL	Set the prefer day
prefer_time	varchar(20)	NOT NULL	Set the prefer time
otp	varchar(20)	NOT NULL	The send the OTP in user Mobile number
status	int(11)	NOT NULL	Update status
name2	varchar(20)	NOT NULL	Add name

mobile2	bigint(20)	NOT NULL	Enter the mobile number
email2	varchar(40)	NOT NULL	Enter the email id
address2	varchar(50)	NOT NULL	Enter the address
req_st	int(11)	NOT NULL	Send the request
deliver_st	int(11)	NOT NULL	Delivered message
sms_st	int(11)	NOT NULL	Send the sms format
reschedule	int(11)	NOT NULL	Again send the reschedule
prefer_st	int(11)	NOT NULL	Set prefer day and time schedules
ration_date	varchar(15)	NOT NULL	Mention the ration shop open date
ration_time	varchar(20)	NOT NULL	Mention ration shop Time

Table 4.5.3 structure for table rq_employee

Field	Type	Constrains	Descriptions
id	int(11)	Primary Key	Admin are create the id
rid	varchar(20)	NOT NULL	This id are created ration id
<i>eid</i>	varchar(20)	NOT NULL	Admin are created the employee id
name	varchar(20)	NOT NULL	Enter the name
address	varchar(50)	NOT NULL	Enter the address
mobile	bigint(20)	NOT NULL	Enter the employee mobile numbers

email	varchar(40)	NOT NULL	Enter the employee email id
pass	varchar(20)	NOT NULL	
name2	varchar(20)	NOT NULL	Enter the next employee name
address2	varchar(50)	NOT NULL	Enter the second employee address
mobile2	bigint(20)	NOT NULL	Enter the second employee mobile number
email2	varchar(40)	NOT NULL	Enter the second employee email id
rdate	varchar(20)	NOT NULL	Ration shop open date announced
status	int(11)	NOT NULL	Update status
given_date	varchar(20)	NOT NULL	Send the schedule date
stime	varchar(20)	NOT NULL	Send the schedule time
etime	varchar(20)	NOT NULL	Enter the employee timing
slot_mode	int(11)	NOT NULL	Define the slot mode
duration	int(11)	NOT NULL	Update the duration

Table 4.5.4 structure for table rq_product

Field	Type	Constrains	Descriptions
id	int(11)	Primary key	Create the product id
product	varchar(50)	NOT NULL	Update the product details

ptype	int(11)	NOT NULL	Mention the protect types
price	int(11)	NOT NULL	Enter the price details
qty	int(11)	NOT NULL	Enter the quantity
qtype	varchar(20)	NOT NULL	Mention the quantity types

Table 4.5.5 structure for table rq_ration

Field	Type	Constrains	Descriptions
id	int(11)	NOT NULL	Create the ration shop id
<i>rid</i>	varchar(20)	Foreign key	Create the unique of the ration id
name	varchar(30)	NOT NULL	Enter the name
rno	varchar(10)	NOT NULL	Enter the ration shop number
building	varchar(20)	NOT NULL	Mention the building
street	varchar(20)	NOT NULL	Mention the street details
area	varchar(20)	NOT NULL	Mention the area details
city	varchar(20)	NOT NULL	Enter the city
district	varchar(20)	NOT NULL	Enter the district
pincode	varchar(20)	NOT NULL	Enter the secrete pincode
phone	varchar(20)	NOT NULL	Enter the mobile number

card_num	int(11)	NOT NULL	Mention or enter the card number
rdate	varchar(20)	NOT NULL	Enter the ration shop open date

Table 4.5.6 structure for table rq_stock

Field	Type	Constrains	Descriptions
id	int(11)	Primary key	Admin are create the id
rid	varchar(20)	NOT NULL	Enter the ration shop id
pid	int(11)	NOT NULL	Enter the product id
product	varchar(30)	NOT NULL	Enter the product details
price	int(11)	NOT NULL	Enter the price
qty	int(11)	NOT NULL	Enter the quantity of the product
qtype	varchar(20)	NOT NULL	Enter the quantity type
rdate	varchar(15)	NOT NULL	Mention the ration shop date
max_qty	int(11)	NOT NULL	Maintain the maximum quantity

Table 4.5.7 structure for table rq_stock_req

Field	Type	Constraints	Descriptions
id	int(11)	Primary key	Admin are create id of the primary key
rid	varchar(20)	NOT NULL	Enter the ration id
product	varchar(100)	NOT NULL	Enter the product details
qty	int(11)	NOT NULL	Mention the product quantity
rdate	varchar(20)	NOT NULL	Update the ration shop open date

Table 4.5.8 structure for table rq_timeslot

Field	Type	Constraints	Descriptions
id	int(11)	Primary key	Create the primary key this also Not null
rid	varchar(20)	NOT NULL	Enter the ration id
given_date	varchar(20)	NOT NULL	Enter the date
stime	varchar(20)	NOT NULL	Enter the schedule time
etime	varchar(20)	NOT NULL	Enter the employee time
duration	int(11)	NOT NULL	Maintain the time duration
break_time	varchar(20)	NOT NULL	Mention the break time
slot_mode	varchar(20)	NOT NULL	Enter the slot mode

slots	text	NOT NULL	Add the slot
rdate	varchar(20)	NOT NULL	Mention the ration date
month	int(11)	NOT NULL	Enter the month of the schedule details
year	int(11)	NOT NULL	Maintain the year of the working schedules

Table 4.5.9 structure for table rq_time_queue

Field	Type	Constrains	Descriptions
id	int(11)	Primary key	Create the primary key
rid	varchar(20)	NOT NULL	Create the ration shop id
card	varchar(20)	NOT NULL	Enter the card details
ration_date	varchar(20)	NOT NULL	Enter the ration shop working dates
ration_time	varchar(20)	NOT NULL	Mention the ration shop working time
deliver_st	int(11)	NOT NULL	Enter the deliver list details
product	varchar(200)	NOT NULL	Enter the product details
amount	int(11)	NOT NULL	Enter the amount details
month	int(11)	NOT NULL	Maintain the monthly product details

year	int(11)	NOT NULL	Maintain the year of the product details
dtime	varchar(20)	NOT NULL	Mention the date and time

4.6 Input Design

Input Design: Input design is **the process of converting user-originated input format to a computer based format**. This computer based format is called as input form or source document.

The screenshot displays the 'Admin Login' interface for 'RATION SHOP'. At the top, a navigation menu includes 'Home', 'Admin' (the active page), 'Employee', and 'Consumer'. The central area features the title 'Admin Login' and the role 'OFFICER'. A prominent green box contains the login form, which consists of a text input field pre-filled with 'admin', a password input field represented by masked dots '*****', and a 'LOGIN' button. The bottom of the page has a dark footer with the text 'Ration Shop'.

Figure 4.1 Admin Login page-Input design

4.7 Output Design

The python output() is **used to print the output for the end-user by taking the input from the keyboard**. The only possibility to print the output is by using the print() statement. There are multiple forms to print this print statement. Printing the print statement without any arguments.

RATION SHOP
Home
Entry
Request
Report
Logout

Report

Date
Month & Year

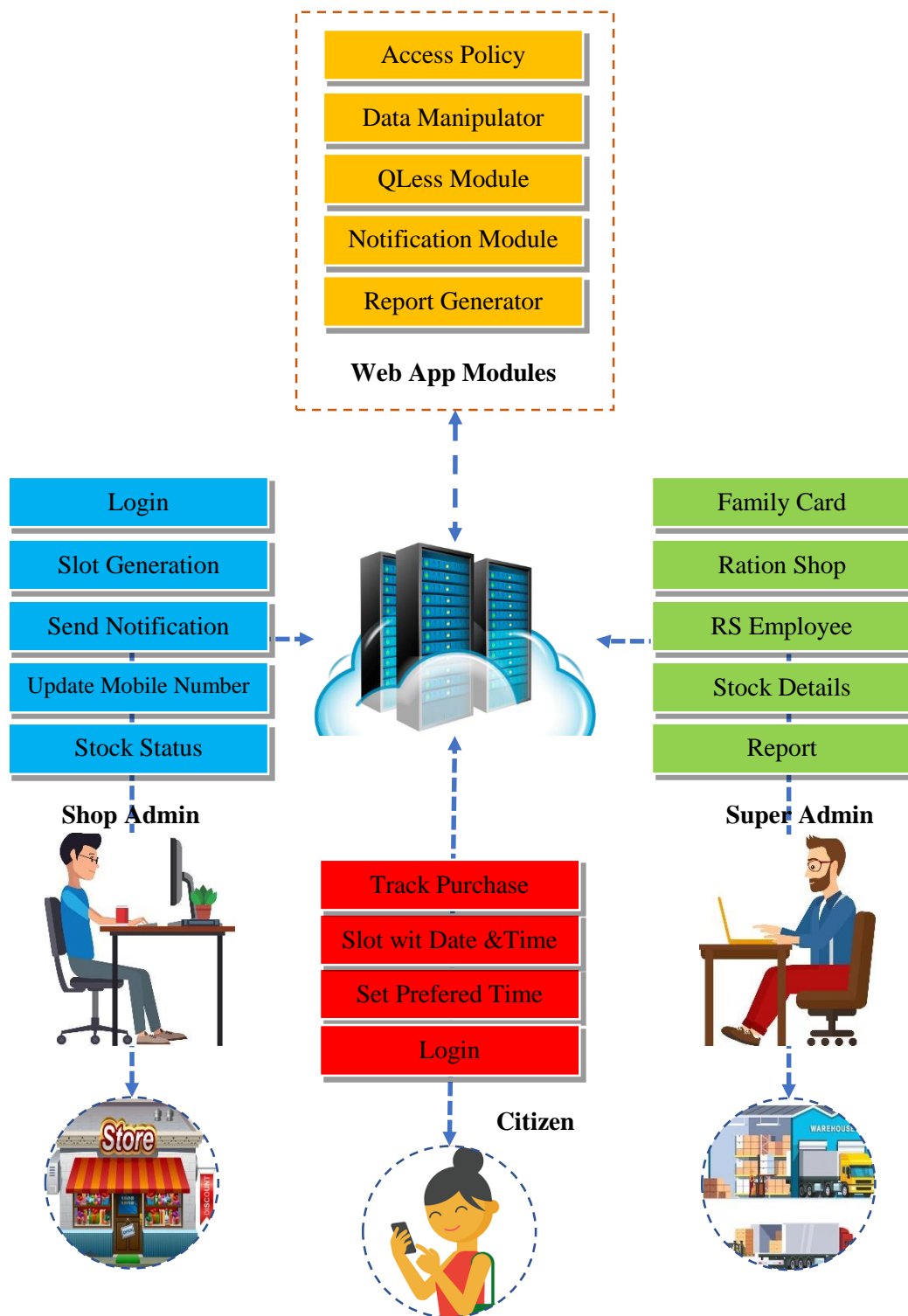
S.No	Date	Time	Duration	Delivered
1	03-01-2022	11:00 - 13:00	10 Minutes	View

Figure 4.2 Report

4.8 Concept Flow

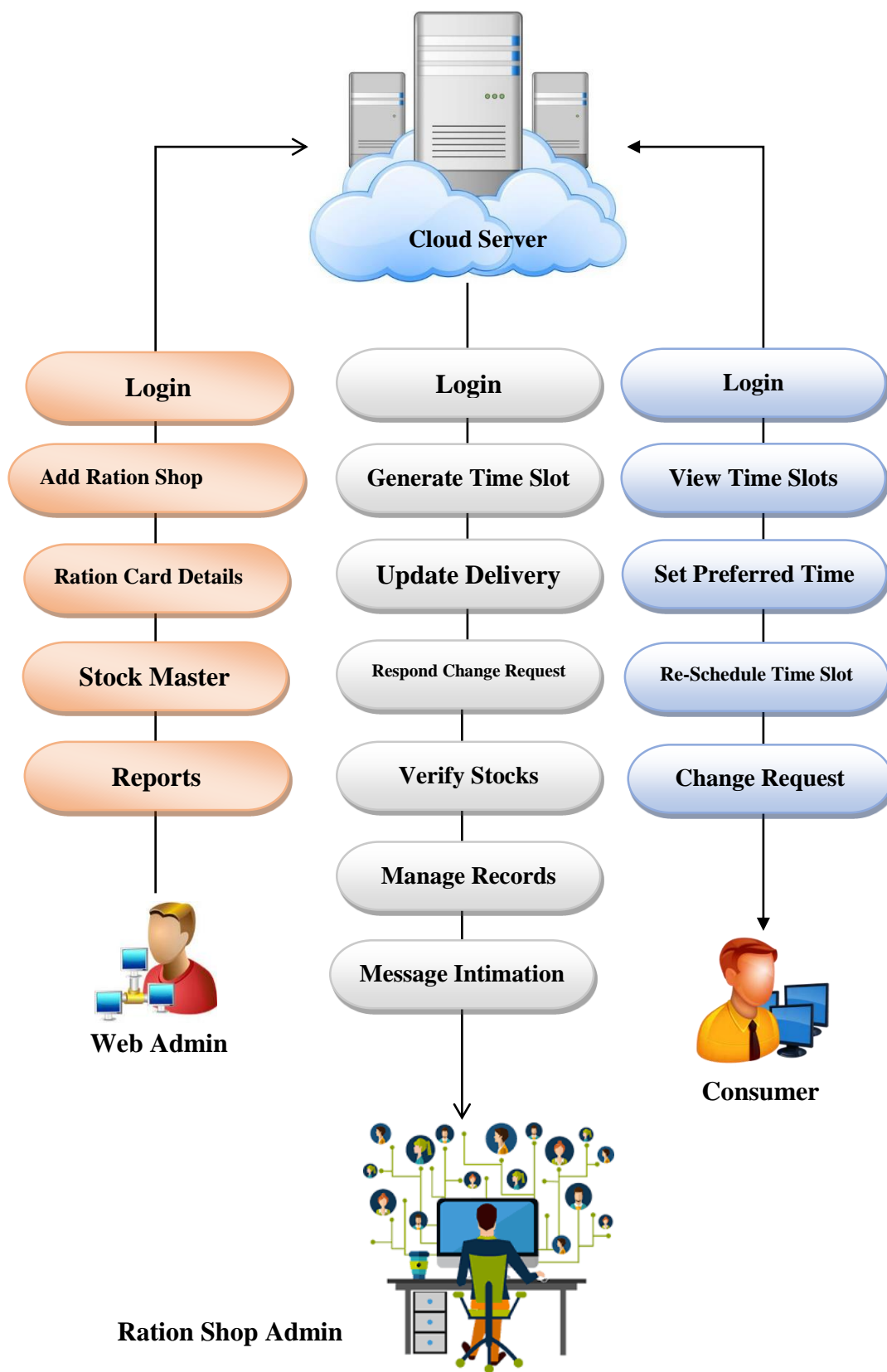


4.9. System Architecture



System Architecture

4.10. System Flow



System Flow Diagram

CHAPTER 5

SYSTEM DEVELOPMENT

5.1 Front End

5.1.1. Python 3.7.4

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.



Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain.

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc. The biggest strength of Python is huge collection of standard libraries which can be used for the following:

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia
- Scientific computing
- Text processing and many more.

5.2 Back End

5.2.1 MySQL

MySQL tutorial provides basic and advanced concepts of MySQL. Our MySQL tutorial is designed for beginners and professionals. MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company. MySQL database that provides for how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc. There are also given MySQL interview questions to help you better understand the MySQL database.



MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications. It is developed, marketed, and supported by MySQL AB, a Swedish company, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is My Ess Que Ell. However, you can pronounce it in your way. Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

5.2.2 The Apache Web Server

In addition to PHP, MySQL, JavaScript, and CSS, there's actually a fifth hero in the dynamic Web: the web server. In the case of this book, that means the Apache web server. We've discussed a little of what a web server does during the HTTP server/client exchange, but it actually does much more behind the scenes. For example, Apache doesn't serve up just HTML files—it handles a wide range of files, from images and Flash files to MP3 audio files, RSS (Really Simple Syndication) feeds, and more. Each element a web client encounters in an HTML page is also requested from the server, which then serves it up. But these objects don't have to be static files, such as GIF images. They can all be generated by programs such as PHP

scripts. That's right: PHP can even create images and other files for you, either on the fly or in advance to serve up later. To do this, you normally have modules either precompiled into Apache or PHP or called up at runtime. One such module is the GD library (short for Graphics Draw), which PHP uses to create and handle graphics.

Apache also supports a huge range of modules of its own. In addition to the PHP module, the most important for your purposes as a web programmer are the modules that handle security. Other examples are the Rewrite module, which enables the web server to handle a varying range of URL types and rewrite them to its own internal requirements, and the Proxy module, which you can use to serve up often-requested pages from a cache to ease the load on the server. Later in the book, you'll see how to actually use some of these modules to enhance the features provided by the core technologies we cover. About Open Source Whether or not being open source is the reason these technologies are so popular has often been debated, but PHP, MySQL, and Apache are the three most commonly used tools in their categories. What can be said, though, is that being open-source means that they have been developed in the community by teams of programmers writing the features they themselves want and need, with the original code available for all to see and change. Bugs can be found and security breaches can be prevented before they happen. There's another benefit: all these programs are free to use. There's no worrying about having to purchase additional licenses if you have to scale up your website and add more servers. And you don't need to check the budget before deciding whether to upgrade to the latest versions of these products.

WampServer

WampServer is a Windows web development environment. It allows you to create web applications with Apache2, PHP and a MySQL database. Alongside, PhpMyAdmin allows you to manage easily your database.



WampServer is a reliable web development software program that lets you create web apps with MYSQL database and PHP Apache2. With an intuitive interface, the application features

numerous functionalities and makes it the preferred choice of developers from around the world. The software is free to use and doesn't require a payment or subscription.

5.2.3 Bootstrap 4

Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.



It solves many problems which we had once, one of which is the cross-browser compatibility issue. Nowadays, the websites are perfect for all the browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones). All thanks to Bootstrap developers -Mark Otto and Jacob Thornton of Twitter, though it was later declared to be an open-source project.

Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap

Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops

Mobile-first approach: In Bootstrap, mobile-first styles are part of the core framework

Browser compatibility: Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera)

5.2.4 Web Framework

Web Application Framework or simply Web Framework represents a collection of libraries and modules that enables a web application developer to write applications without having to bother about low-level details such as protocols, thread management etc.

Using an IDE

As good as dedicated program editors can be for your programming productivity, their utility pales into insignificance when compared to Integrated Developing Environments (IDEs), which offer many additional features such as in-editor debugging and program testing, as well as function descriptions and much more.

Flask

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.



Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have formed a validation support. Instead, Flask supports the extensions to add such functionality to the application. Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers. Let's take a closer look into Flask, so-called "micro" framework for Python.

Flask was designed to be easy to use and extend. The idea behind Flask is to build a solid foundation for web applications of different complexity. From then on you are free to plug in any extensions you think you need. Also, one can free to build their modules. Flask is great for all kinds of projects. It's especially good for prototyping.

Flask is part of the categories of the micro-framework. Micro-framework is normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins. In the case of Flask, its dependencies are: **WSGI**-Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.

Werkzeug-It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

Jinja2 Jinja2 is a popular templating engine for Python. A web templating system combines a template with a certain data source to render dynamic web pages.

CHAPTER 6

SYSTEM TESTING & IMPLEMENTATION

6.1 Valid Payloads

Scenario 1: Should be able to register user with valid registration request payload

Payload:

- All fields
- All required fields

Check:

- API response status code: 200
- Database: User should exist in database

Invalid Payloads

Scenario 2: Should not be able to Register user with QLessPDS Dashboard

Payload:

- Missing/Empty Headers
- Missing/Empty Required Fields
- Incorrect values/Incorrect format for Required Fields
- Various combinations of invalid email formats

Check:

- API response status code: 400
- Database: No record is created in DB

6.1.1 Verify User Registration

Quite often, when a user is registered, a verification link is sent to the user's email address to ensure the email is valid. Users normally need to click the link in the email to verify their email account. We can test with valid and invalid verification link.

Scenario 3: Verify user registration

Payload:

- Valid request

Check:

- API response status code: 200
- Database: Check user status changes from unverified to verified

Scenario 4: Verify user registration

Payload:

- Invalid request

Check:

- API response status code: 400 or 403
- Database: Check user status doesn't change. User should still be unverified.
- Login: Check user cannot login if status is unverified.

6.1.2 Re-registration

Scenario 5: Should not be able to register the same user twice

Payload:

- Valid request

Check:

- API response status code: Depends
- Response message: Error message saying user already exists. Note: for security reasons, some applications may not return this message.

6.1.3 Restricted Passwords

Restricted passwords are the ones that have already been hacked and put on paste bins. HIBP (HaveIBeenPawnd.com) has a list of hacked passwords.

Scenario 6: Should not be able to register user with restricted passwords

Payload:

- All fields valid, but restricted password

Check:

- API response status code: 400
- Response message: Check error message is appropriate

6.1.4 Easy to Guess Passwords

There is a list of passwords that are common and easy to hack so should be avoided in user registration.

Scenario 7: Should not be able to register user with easy to guess passwords

Payload:

- All fields valid, but easy to guess password

Check:

- API response status code: 400
- Response message: Check error message is appropriate

6.1.5 Password Same as Username

Scenario 8: Should not be able to set password the same as username

Payload:

- All fields valid, but password same as username

Check:

- API response status code: 400
- Response message: Check error message is appropriate

6.2 Unit Testing

Unit testing is a method of determining the correctness of a single function isolated from a larger codebase. The idea is that if all the atomic units of an application work as intended in isolation, then integrating them together as intended is much easier.

Unit testing tools

There are many tools for creating tests in Python. Some of these tools, such as pytest, replace the built-in unittest framework. Other tools, such as nose, are extensions that ease test case creation. Note that many of these tools are also used for integration testing by writing the test cases to exercise multiple parts of code at once.

- Unit test is the built-in standard library tool for testing Python code.
- Pytest is a complete testing tool that emphasizes backwards-compatibility and minimizing boilerplate code.
- Nose is an extension to unittest that makes it easier to create and execute test cases.
- Hypothesis is a unit test-generation tool that assists developers in creating tests that exercise edge cases in code blocks. The best way to get started using Hypothesis is by going through the well-written quickstart.
- Mimesis generates synthetic test data which is useful to apply in your tests.
- Testify was a testing framework meant to replace the common unittest+nose combination. However, the team behind testify is transitioning to pytest so it's recommended you do not use testify for new projects.

- **Data centric**

Data centric refers to an architecture where data is the primary and permanent asset, and applications come and go. In the data centric architecture, the data model precedes the implementation of any given application and will be around and valid long after it is gone.

Many people may think this is what happens now or what should happen. But it very rarely happens this way. Businesses want functionality, and they purchase or build application systems. Each application system has its own data model, and its code is inextricably tied with this data model. It is extremely difficult to change the data model of an implemented application system, as there may be millions of lines of code dependent on the existing model.

- **Functionality Testing**

FUNCTIONAL TESTING is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

- **Integration Testing**

Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated

- **Boundary Testing**

Boundary testing is **the process of testing between extreme ends or boundaries between partitions of the input values**. So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called “boundary testing”

Boundary Value Testing

AGE * Accepts Value 21 to 65

Boundary Value Test Case		
Invalid Test Case (Min value - 1)	Valid Test Case (Min + Min, Max, -Max)	Invalid Test Case (Max value - 1)
20	21,22,65,64	66

www.educba.com

Figure 6.1 Boundary Testing

○ **Test cases**

S.NO	SCENARIO	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT
1	Ration Admin login form	Username & Password	Login	Login success
2	Add Ration Employee	Ration Employee details	Update Successful	Ration Employee details stored database
3	Ration employee login form	Username & Passwords	Login	Login Success
4	Add Peoples details	People details	Update Successful	Peoples details stored database
5	Add Time Schedule details	Time Schedule details	Update successful	Time Schedule details stored database
6	Predefined time	Predefined Time	Set & Request Predefined time	Stored Predefined time and Set Schedule

SYSTEM IMPLEMENTATION

6.3 Purpose

The purpose of System Implementation can be summarized as follows: making the new system available to a prepared set of users (the deployment), and positioning on-going support and maintenance of the system within the Performing Organization (the transition). At a finer level of detail, deploying the system consists of executing all steps necessary to educate the Consumers on the use of the new system, placing the newly developed system into production, confirming that all data required at the start of operations is available and accurate, and validating that business functions that interact with the system are functioning properly. Transitioning the system support responsibilities involves changing from a system development to a system support and maintenance mode of operation, with ownership of the new system moving from the Project Team to the Performing Organization.

The implementation phase uses the most project time and resources, and as a result, costs are usually the highest during this phase. Project managers also experience the greatest conflicts over schedules in this phase. You may find as you are monitoring your project that the actual time it is taking to do the scheduled work is longer than the amount of time planned.

When you absolutely have to meet the date and you are running behind, you can sometimes find ways to do activities more quickly by adding more resources to critical path tasks. That's called *crashing*. Crashing the schedule means adding resources or moving them around to bring the project back into line with the schedule. Crashing **always** costs more and doesn't always work. There's no way to crash a schedule without raising the overall cost of the project. So, if the budget is fixed and you don't have any extra money to spend, you can't use this technique.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1. Conclusion

Physically queueing is a reality on many industries that provide services or sell goods. Waiting in a queue can be stressful and exhausting for the clients because of the enforced idle time, and may lead to decreased customer satisfaction. In this project, we have explored how reinforcement learning can be used for predicting the slot with time of people in fair Price Shop. We presented QLess using Deep Q-Learning based web application that allows FPS admin to create a virtual queue, and notify the citizen. Virtual queueing system for ration center website is developed to overcome the uncertainties in ration centers. This system will avoid the corruption in ration system to a larger extent by providing transparency to users at each level. This system will be helpful to save time and efforts of standing in long waiting queues, the application will also be helpful for organizations to serve better to the customers without making them wait in queues this in turn can boost profit and increase the quality of Service. This system can be successfully implemented in environment where crowd management is difficult and thus help in the elimination of physical lines and waiting time all over the country in service-based institutions and organizations.

7.2. Future Enhancement

Perhaps, in future we will try our best to develop an option of money payment for ration instead of paying at FPS to control corruption and loss to the government.

CHAPTER 8

APPENDICES

8.1 Source code

Add Ration Shop Employee

```
@app.route('/add_emp',methods=['POST','GET'])
def add_emp():
    result=""
    act=""
    rid = request.args.get('rid')
    if request.method=='POST':
        name=request.form['name']
        rid=request.form['rid']
        mobile=request.form['mobile']
        email=request.form['email']
        address=request.form['address']

        now = datetime.datetime.now()
        rdate=now.strftime("%d-%m-%Y")
        mycursor = mydb.cursor()

        mycursor.execute("SELECT max(id)+1 FROM rq_employee")
        maxid = mycursor.fetchone()[0]
        if maxid is None:
            maxid=1

        eid="E"+str(maxid)
        pass1="1234"
        sql = "INSERT INTO rq_employee(id, rid, eid, name, address, mobile, email, pass,
rdate, status) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"
        val = (maxid, rid, eid, name, address, mobile, email, pass1, rdate, '1')
        print(sql)
        mycursor.execute(sql, val)
        mydb.commit()
        print(mycursor.rowcount, "record inserted.")
        act='success'
        return redirect(url_for('add_assistant',sid=maxid))
        #if mycursor.rowcount==1:
        #    result="Registered Success"

    return render_template('add_emp.html',act=act,rid=rid)
```

Deep Q Learning

```
def findTime(T, K):

    # convert the given time in minutes
    minutes = (((ord(T[0]) - ord('0'))* 10 +
```

```

        ord(T[1]) - ord('0'))* 60 +
        ((ord(T[3]) - ord('0')) * 10 +
        ord(T[4]) - ord('0')));

# Add K to current minutes
minutes += K

# Obtain the new hour
# and new minutes from minutes
hour = (int(minutes / 60)) % 24

min = minutes % 60
hh=""
mm=""
# Print the hour in appropriate format
if (hour < 10):
    hh="0"+str(hour)
    #print(0, hour, ":", end = "")

else:
    hh="" + str(hour)
    #print(hour, ":", end = "")

# Print the minute in appropriate format
if (min < 10):
    mm="0"+str(min)
    #print(0, min, end = "")

else:
    mm="" + str(min)
    #print(min, end = "")
hm=hh+": "+mm
return hm

def findDay(date):
    born = datetime.datetime.strptime(date, '%d-%m-%Y').weekday()
    return (calendar.day_name[born])

@app.route('/deep_q_learning', methods=['POST', 'GET'])
def deep_q_learning():
    act=""
    uname=""
    c1=""
    c2=""
    c3=""
    available=""
    data3=[]
    if 'username' in session:
        uname = session['username']

```

```

ff=open("usr.txt","r")
uu=ff.read()
ff.close()
if uname is None:
    uname=uu

mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM rq_employee where eid=%s",(uname, ))
dt = mycursor.fetchone()
rid=dt[1]
mycursor.execute("SELECT * FROM rq_ration where rid=%s",(rid, ))
data = mycursor.fetchone()

mycursor.execute("SELECT * FROM rq_stock where rid=%s",(rid, ))
data1 = mycursor.fetchall()

if request.method=='POST':
    rdate=request.form['rdate']
    stime=request.form['stime']
    etime=request.form['etime']
    duration=request.form['duration']
    users=request.form['users']
    stype=request.form['stype']

    stm=stime.split(':')
    etm=etime.split(':')

    try:
        c1=request.form['c1']
    except:
        print("")
    try:
        c2=request.form['c2']
    except:
        print("")
    try:
        c3=request.form['c3']
    except:
        print("")
    gd=rdate.split("-")
    gdate=gd[2]+"-"+gd[1]+"-"+gd[0]

    day1=findDay(gdate)
    print(day1)

#####
ui=1
q_user=int(users)
q_length=q_user
user_count=0

```

```

arru=[]
arr_u=[]
data2=[]
divide=0

```

```

mycursor.execute("UPDATE rq_consumer SET sms_st=0 WHERE rid=%s",(rid,))
mydb.commit()

```

```

if stype=="0":
    mycursor.execute("SELECT count(*) FROM rq_consumer where status=0 &&
prefer_st=0 && rid=%s",(rid, ))
    user_count = mycursor.fetchone()[0]
    divide=int(user_count/q_user)

    mycursor.execute("SELECT * FROM rq_consumer where status=0 && prefer_st=0
&& rid=%s",(rid, ))
    data2 = mycursor.fetchall()

```

```

elif stype=="1":

    mycursor.execute("SELECT * FROM rq_consumer where status=0 && prefer_st=1
&& rid=%s",(rid, ))
    tot_consumer = mycursor.fetchall()
    for dds in tot_consumer:
        ex=0
        ey=0
        ex1=dds[7].split(',')
        for ex2 in ex1:
            if ex2==day1:
                ex+=1
        ex3=dds[8].split('-')
        sx=ex3[0].split(':')
        sy=ex3[1].split(':')
        if sx[0]<=stm[0] or sy[0]<=etm[0]:
            ey+=1
        if ex>0 and ey>0:
            user_count+=1
            mycursor.execute("SELECT * FROM rq_consumer where card=%s",(dds[3], ))
            cdd = mycursor.fetchone()
            data2.append(cdd)
        divide=int(user_count/q_user)

```

```

elif stype=="2":
    mycursor.execute("SELECT count(*) FROM rq_consumer where status=1 &&
deliver_st=0 && rid=%s",(rid, ))
    user_count = mycursor.fetchone()[0]
    divide=int(user_count/q_user)

```

```

        mycursor.execute("SELECT * FROM rq_consumer where status=1 && deliver_st=0
&& rid=%s",(rid, ))
        data2 = mycursor.fetchall()

        i=0
        for ur in data2:

            arr_u.append(ur[3])
            if i==divide:
                arru.append(arr_u)
            if ui==q_length:
                arru.append(arr_u)
                arr_u=[]
                q_length+=q_user

            i+=1

        ui+=1

#####

time=time
ee=etime.split(":")
e1=int(ee[0])
e2=int(ee[1])
i=0
ii=0
k=int(duration)
K=k
rk=0
T=time
act="1 "
s=""
ss=""
no_of_customer=0
while i<41:
    dat1=[]
    s="1 "
    tm1=time
    time=findTime(T, K)
    tm2=time
    tt=tm2.split(":")
    t1=int(tt[0])
    t2=int(tt[1])
    tm=tm1+" to "+tm2

    if c1=="1":
        if t1==12 and t2<15:

            rk=15-k

```


K+=rk

```
if tm=="12:00 to 12:15":
    s="2"
elif tm=="12:10 to 12:25":
    s="2"
```

```
if c2=="2":
    if t1==14 and t2<10:
        rk=30-k
        K+=rk
    elif t1==14 and t2<20:
        rk=30-k
        K+=rk
```

```
if tm=="14:00 to 14:30":
    s="3"
elif tm=="14:05 to 14:35":
    s="3"
elif tm=="14:10 to 14:40":
    s="3"
```

```
if c3=="3":
    if t1==15 and t2>=30 and t2<45:
        rk=15-k
        K+=rk
```

```
if tm=="15:30 to 15:45":
    s="2"
elif tm=="15:35 to 15:50":
    s="2"
elif tm=="15:40 to 15:55":
    s="2"
```

K+=k

```
if t1>=e1 and t2>=e2:
    print("yes")
    tm=tm1+" to "+str(e1)+":"+str(e2)
    dat1.append(tm)
    dat1.append(s)
    break
```

```

    dat1.append(tm)
    dat1.append(s)

    if ii<divide:
        dat1.append(arru[ii])

        for ss in arru[ii]:
            no_of_customer+=1
            mycursor.execute("UPDATE rq_consumer SET
ration_date=%s,ration_time=%s,sms_st=1 WHERE card=%s",(gdate,tm,ss))
            mydb.commit()
        else:
            arr_empty=["1"]
            dat1.append(arr_empty[0])

    if s=="2" or s=="3":
        print("")
    else:
        ii+=1

    data3.append(dat1)
    i+=1

    print("No. of Customers:"+str(no_of_customer))
    mycursor.execute("SELECT * FROM rq_stock where rid=%s",(rid, ))
    data4 = mycursor.fetchall()
    n_stock=0

    for rw in data4:
        tot=rw[5]
        deliver_stock=rw[8]*no_of_customer
        print("stock="+str(deliver_stock))
        if deliver_stock<=tot:
            n_stock+=1

    if n_stock>0:
        mycursor.execute("UPDATE rq_employee SET
given_date=%s,stime=%s,etime=%s,slot_mode=%s,duration=%s WHERE
eid=%s",(gdate,stime,etime,stype,duration,uname))
        mydb.commit()
        available="yes"
    else:
        available="no"

    return
    render_template('deep_q_learning.html',data=data,data1=data1,data3=data3,act=act,available
=available)

```

Ration Shop – Billing

```

@app.route('/emp_entry',methods=['POST','GET'])
def emp_entry():
    act=""
    msg=""
    uname=""
    data2=[]
    data4=[]
    gtime=""
    act = request.args.get('act')
    card = request.args.get('card')
    tid = request.args.get('tid')

    if 'username' in session:
        uname = session['username']

    now = datetime.datetime.now()
    rdate=now.strftime("%d-%m-%Y")
    rh=now.strftime("%H")
    rm=now.strftime("%M")
    dtime=rh+":"+rm
    mycursor = mydb.cursor()
    mycursor.execute("SELECT * FROM rq_employee where eid=%s",(uname, ))
    dt = mycursor.fetchone()
    rid=dt[1]

    mycursor.execute("SELECT * FROM rq_timeslot where given_date=%s &&
rid=%s",(rdate,rid))
    data = mycursor.fetchall()

    mycursor.execute("SELECT * FROM rq_time_queue where ration_date=%s &&
rid=%s",(rdate,rid))
    data3 = mycursor.fetchall()
    for rs3 in data3:
        tt=rs3[5]
        tm1=tt.split(' to ')
        tm2=tm1[0]
        #print(tm2)
        tm3=tm1[1]

        time_h1=tm2.split(':')
        time_h2=tm3.split(':')

        hh1=time_h1[0]
        mm1=time_h1[1]
        hh2=time_h2[0]
        mm2=time_h2[1]

    if hh1<=rh and mm1<rm and hh2>=rh and mm2>=rm:

```

```

        print(tt)
        gtime=tt

    mycursor.execute("SELECT card FROM rq_time_queue where deliver_st=0 &&
ration_date=%s && rid=%s && ration_time=%s",(rdate,rid,gtime))
    dat1 = mycursor.fetchall()
    for rc in dat1:
        dat3=[]
        mycursor.execute("SELECT name FROM rq_consumer where card=%s",(rc[0], ))
        dat2 = mycursor.fetchone()[0]
        dat3.append(rc[0])
        dat3.append(dat2)
        data4.append(dat3)

    mycursor.execute("SELECT * FROM rq_stock where qty>0 && rid=%s",(rid, ))
    data5 = mycursor.fetchall()

    if tid is not None:
        mycursor.execute("SELECT * FROM rq_time_queue where tid=%s",(tid, ))
        data2 = mycursor.fetchall()

    sk=0
    stock_arr=[]
    sarr=[]
    amount=0
    amt=0
    if request.method=='POST':
        for rd5 in data5:

            sid=rd5[0]
            qt=request.form['qt'+str(sid)]
            qtt=int(qt)
            if qtt>0:
                if rd5[5]>=qtt:
                    print(rd5[3])
                    ss=rd5[3]+"-"+str(qtt)
                    sarr.append(ss)

                    amt=qtt*rd5[4]
                    amount+=amt
                    mycursor.execute("UPDATE rq_stock SET qty=qty-%s WHERE rid=%s &&
id=%s",(qtt,rid,sid))
                    mydb.commit()

                else:
                    sk+=1
    print(sarr)

```

```

print("Amt="+str(amount))
if sk==0:
    prd=",".join(sarr)
    mycursor.execute("UPDATE rq_time_queue SET
deliver_st=1,product=%s,amount=%s,dtime=%s WHERE ration_date=%s && rid=%s &&
card=%s",(prd,amount,dtime,rdate,rid,card))
    mydb.commit()
    mycursor.execute("UPDATE rq_consumer SET deliver_St=1 WHERE rid=%s &&
card=%s",(rid,card))
    mydb.commit()

    print("success")
    msg="Bill Success - "+prd+", Amount: Rs."+str(amount)
else:
    msg="Stock not available"

return
render_template('emp_entry.html',msg=msg,data=data,data2=data2,gtime=gtime,data4=data4
,data5=data5,card=card,act=act,tid=tid)

```

8.2 Screenshot

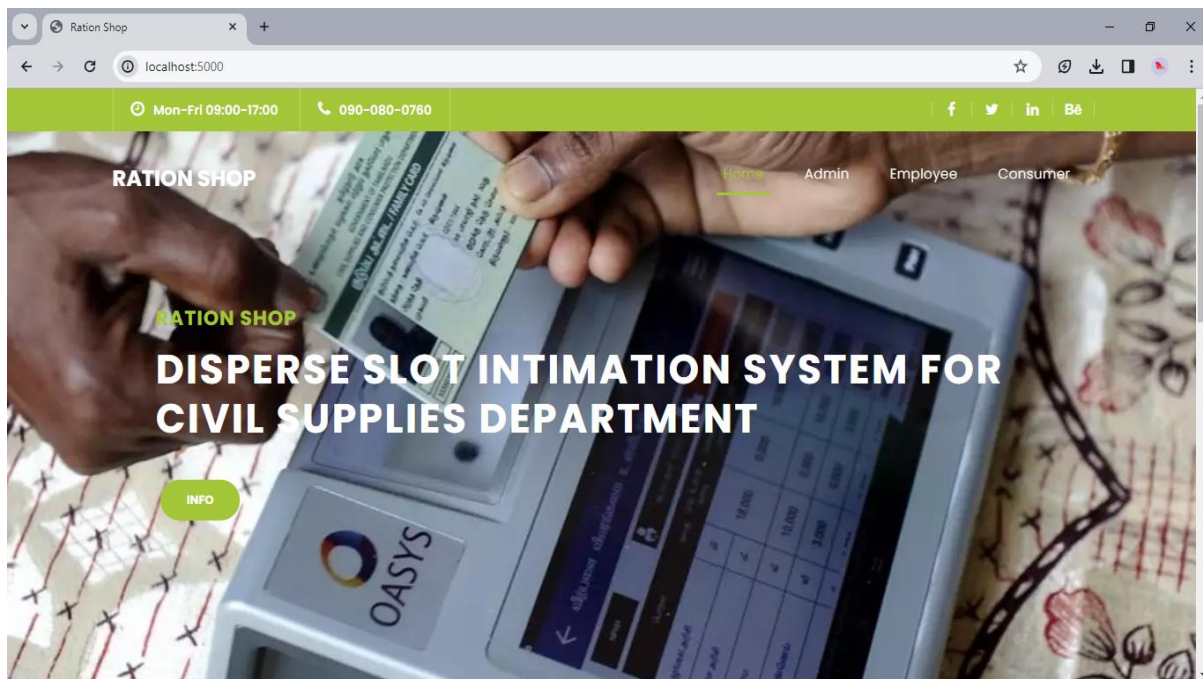
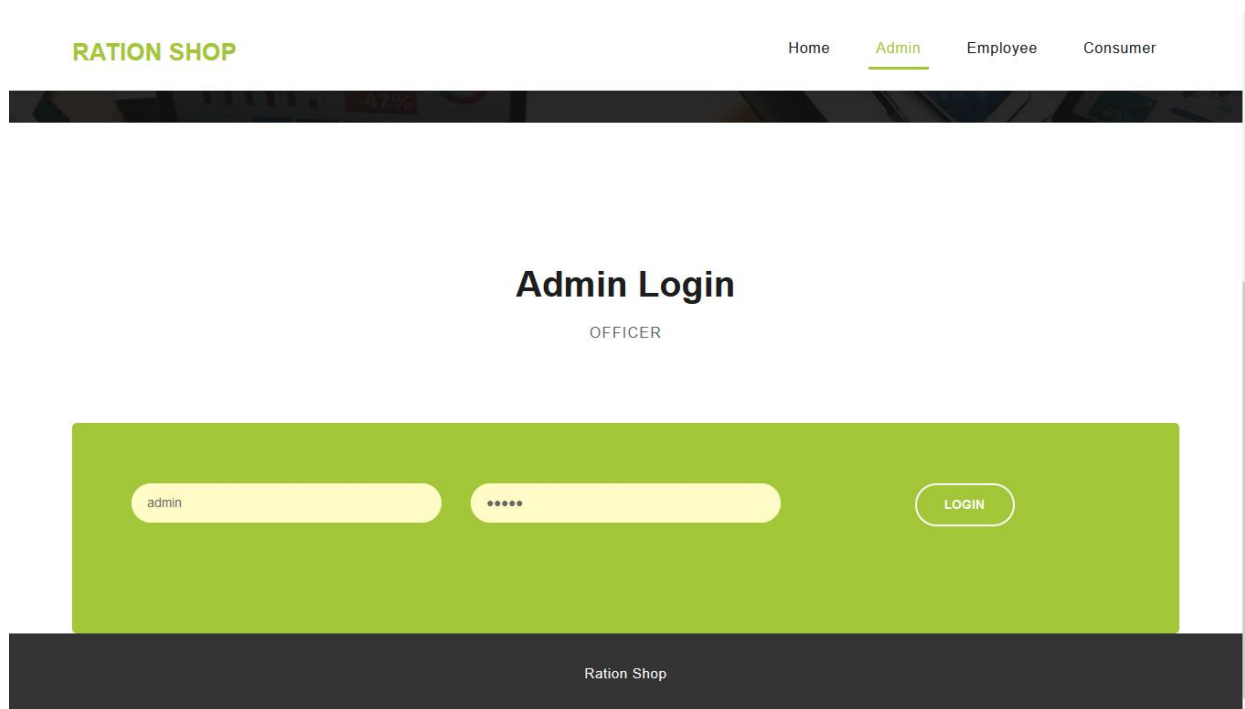


Figure 8.2.1 Form page



RATION SHOP

Home Admin Employee Consumer

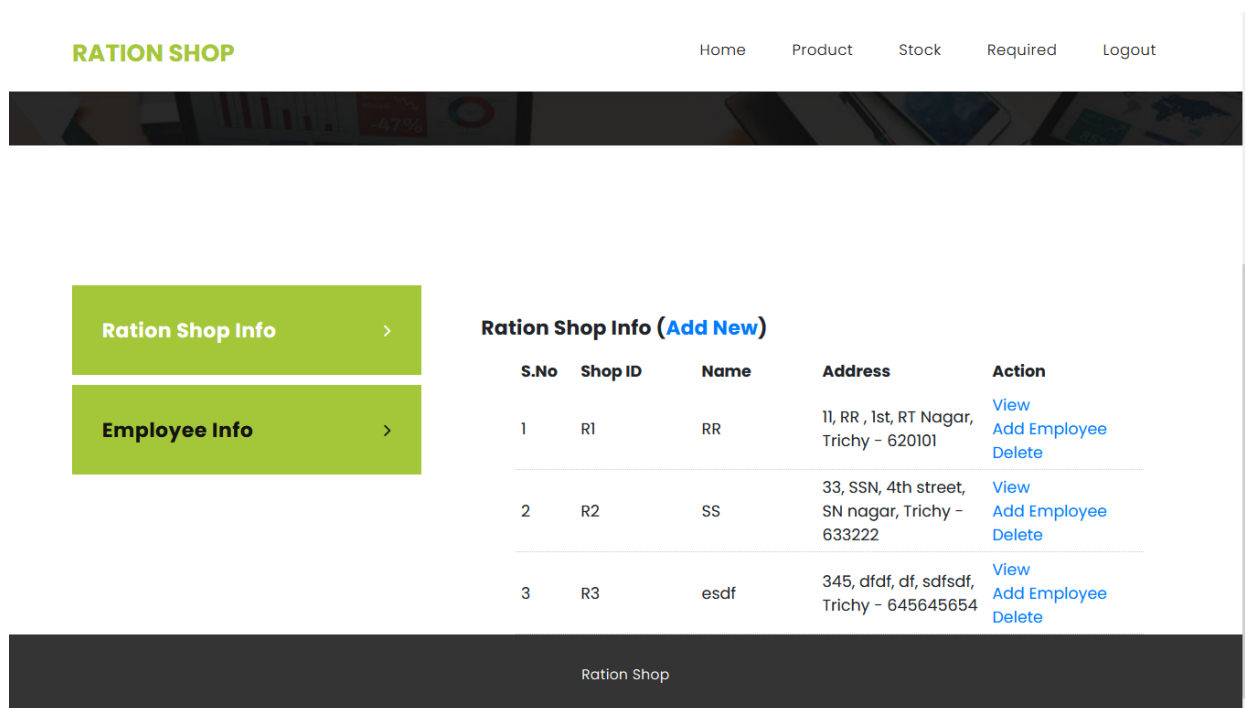
Admin Login

OFFICER

admin ***** LOGIN

Ration Shop

Figure 8.2.2 Admin Login page



RATION SHOP

Home Product Stock Required Logout

Ration Shop Info >

Employee Info >

Ration Shop Info ([Add New](#))

S.No	Shop ID	Name	Address	Action
1	R1	RR	11, RR , 1st, RT Nagar, Trichy - 620101	View Add Employee Delete
2	R2	SS	33, SSN, 4th street, SN nagar, Trichy - 633222	View Add Employee Delete
3	R3	esdf	345, dfdf, df, sdfsd, Trichy - 645645654	View Add Employee Delete

Ration Shop

Figure 8.2.3 Add Ration shop informations

RATION SHOP
Home
Logout

Ration Shop
ADD NEW

Ration Shop Information

Ration Shop Name

Ration Shop Address - No.

Building Name

Street

Area

City

Pincode

Ration Shop Ph Number

ADD

Back

Figure 8.2.4 Ration shop

RATION SHOP
Home
Logout

SHOP INFORMATION

Ration Shop Info

Ration Shop ID : R1

Ration Shop Name : RR

Address : 11, RR , 1st

Area : RT Nagar

City : Trichy

Pincode : 620101

Phone Number : 0431-2321232

No. of Card : 50

Add Card / Card Details

Ration Shop

Figure 8.2.5 View Ration shop details

RATION SHOP [Home](#) [Logout](#)

Card Allotment

No. of Cards

No file selected.

Ration Shop

Figure 8.2.6 Card allotment

RATION SHOP [Home](#) [Logout](#)

Admin

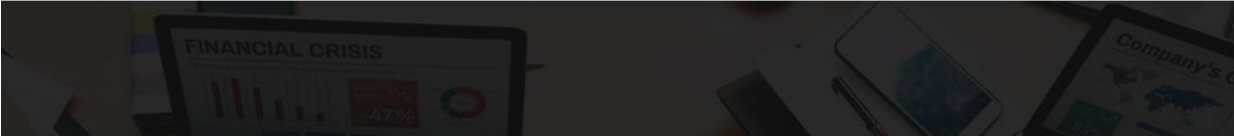
SHOP INFORMATION

Consumer Info

S.No	Name	Card No.	Mobile No.	Action
1	Siva	246301	9976572211	Delete
2	Sathish	246302	8664357421	Delete
3	Murugan	246303	9976573303	Delete
4	Dinesh	246304	9864543342	Delete
5	Raj	246305	9034583454	Delete
6	Kumar	246306	8623424434	Delete
7	Gopi	246307	7578789432	Delete
8	Suresh	246308	9453453554	Delete

Figure 8.2.7 Admin add the consumer informations

RATION SHOP Home Product Stock Required Logout



Ration Shop Info >

Employee Info >

Employee Info

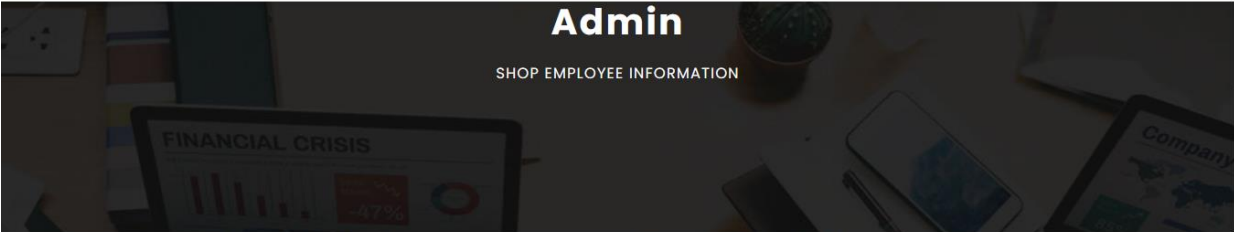
(Shop Admin)

S.No	Name	Address	Mobile No.	E-mail	Action
1	Ramesh	34,FG Colony, Trichy	9956744242	ramesh@gmail.com	View / Delete
2	Rahul	33,ggg	9867845834	rahul3@gmail.com	View / Delete

Ration Shop

Figure 8.2.8 Add Employee informations

RATION SHOP Home Logout



Admin

SHOP EMPLOYEE INFORMATION

Shop Admin Info

Ration Shop ID : R1
Employee ID : E1
Name : Ramesh
Address : 34,FG Colony, Trichy
Mobile No. : 9956744242
E-mail : ramesh@gmail.com
[Edit](#)

Shop Assistant Info

Name : Ravi
Address : 3/11, KG Colony, Salem
Mobile No. : 8854632401
E-mail : ravi@gmail.com
[Edit](#)

Ration Shop

Figure 8.2.9 Admin add shop admin and shop assistant informations

RATION SHOP
Home
Logout

Ration Products

S.No	Product	Given by	Action
1	Rice	Free	Delete
2	Wheat	Free (Quantity: 2 Kg)	Delete
3	Sugar	Price: Rs. 40 (Quantity: 2 Kg)	Delete
4	Oil	Price: Rs. 30 (Quantity: 1 Kg)	Delete

Ration Shop

Figure 8.2.10 Ration Product details(a)

RATION SHOP
Home
Logout

Ration Products

S.No	Product	Given by	Action
1	Rice	Free	Delete
2	Wheat	Free (Quantity: 2 Kg)	Delete
3	Sugar	Price: Rs. 40 (Quantity: 2 Kg)	Delete
4	Oil	Price: Rs. 30 (Quantity: 1 Kg)	Delete

Ration Shop

Figure 8.2.11 Ration shop product details(b)

RATION SHOP

Home Admin Employee Consumer

Shop Admin Login

RATION SHOP

E1

LOGIN

Ration Shop

Figure 8.2.12 Ration shop admin login page

RATION SHOP

Home Entry Request Report Logout

ID: R1, Ration Shop: RR

11, RR, 1st, Rt Nagar, Trichy - 620101

Date

dd / mm / yyyy

Time Interval

-Start Time-

-End Time-

Slot Duration

10 Minutes

Break Timings

☐ Break (12:00-12:15PM)

☐ Lunch (2:00-2:30PM)

☐ Break (3:45-4:00PM)

Slot Mode

Preferred

Generate Time Slot Clear

Figure 8.2.13 Ration shop time schedule

RATION SHOP
Home
Product
Stock
Required
Logout

Deliver Stocks

Ration Shop
R1 - RR

Product
Rice

Maximum Quantity for Each Customer
20

Quantity
2000

Select here
Kg

SUBMIT

Ration Shop

Figure 8.2.14 Deliver stocks

RATION SHOP
Home
Product
Stock
Required
Logout

Stocks - Delivered to Ration Shop

[Stock Deliver](#)

-Ration Shop-

SUBMIT

S.No	Product	Available	Max. Deliver	Date	Action
1	Rice	2500 Kg	20	30-12-2021	Add Stock / Delete
2	Wheat	200 Kg	2	02-01-2022	Add Stock / Delete
3	Sugar	200 Kg	2	02-01-2022	Add Stock / Delete
4	Oil	100 Kg	1	02-01-2022	Add Stock / Delete

Ration Shop

Figure 8.2.15 Stocks delivered to ration shop

ID: R1, Ration Shop: RR

11, RR , 1st, RT Nagar, Trichy - 620101

Available Products

Product	Quantity
Rice	2000 Kg

Date

01 / 01 / 2022

Time Interval

11.30 AM

4.30 PM

Slot Duration

10 Minutes

3

Break Timings



Break (12:00-12:15PM)



Lunch (2:00-2:30PM)



Break (3:45-4:00PM)

Slot Mode

Normal

Generate Time Slot

Figure 8.2.16 available product details



11:30 to 11:40

246301

246302

246303



11:40 to 11:50

246304

246305

246306



11:50 to 12:00

246307

246308

246309



12:00 to 12:15



12:15 to 12:25

246310

246311

246312



12:25 to 12:35

246313

246314

246315



12:35 to 12:45

246316

246317

246318



12:45 to 12:55

246319

246320

246321



12:55 to 13:05

246322

246323

246324

Figure 8.2.17 Time schedule the consumer(a)



Figure 8.2.18 Time schedule the consumer(b)

Available Products

Product	Quantity
Rice	2500 Kg
Wheat	200 Kg
Sugar	200 Kg
Oil	100 Kg

NEW SCHEDULE

AVAILABLE STOCKS

ID: R1, Ration Shop: RR
11, RR, 1st, RT Nagar, Trichy - 620101

Date: 03 / 01 / 2022

Time Interval: 11 AM to 3 PM

Slot Duration: 10 Minutes to 2

Break Timings: ☐ Break (12:00-12:15PM) ☐ Lunch (2:00-2:30PM) ☐ Break (3:45-4:00PM)

Slot Mode: Normal

Generate Time Slot

Figure 8.2.19 Time Reschedule the consumer(a)

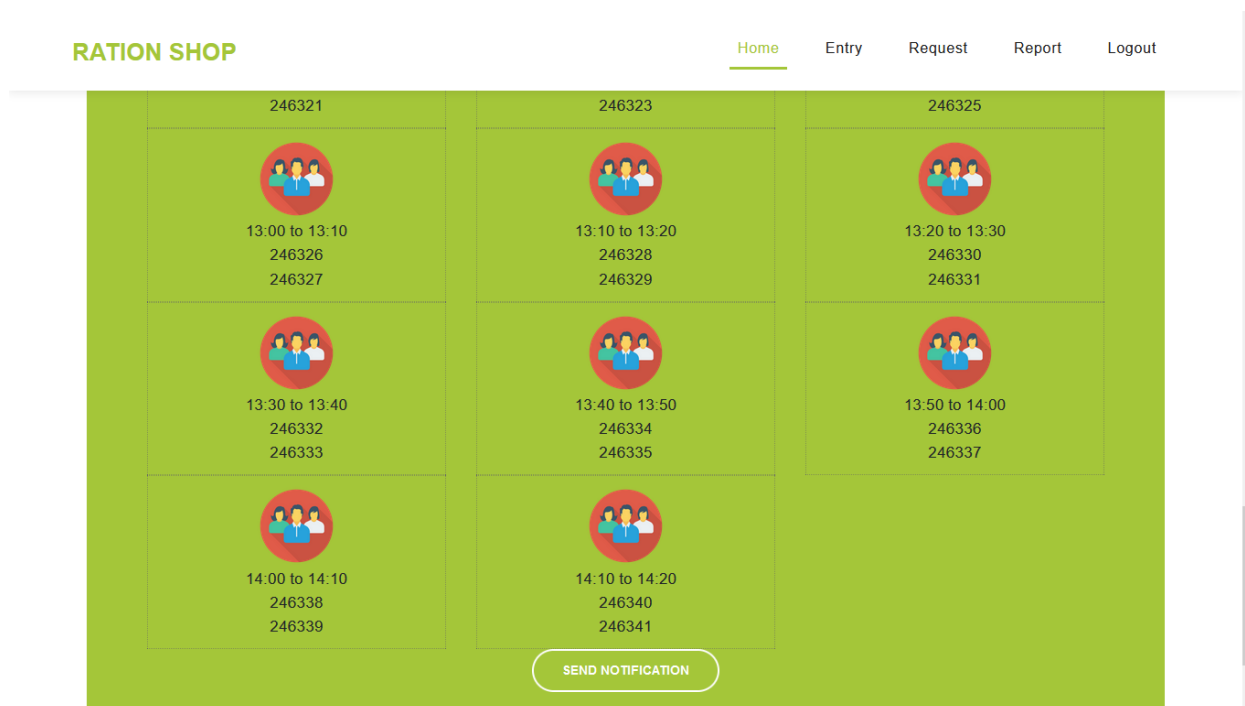


Figure 8.2.20 Time Reschedule the consumer(b)

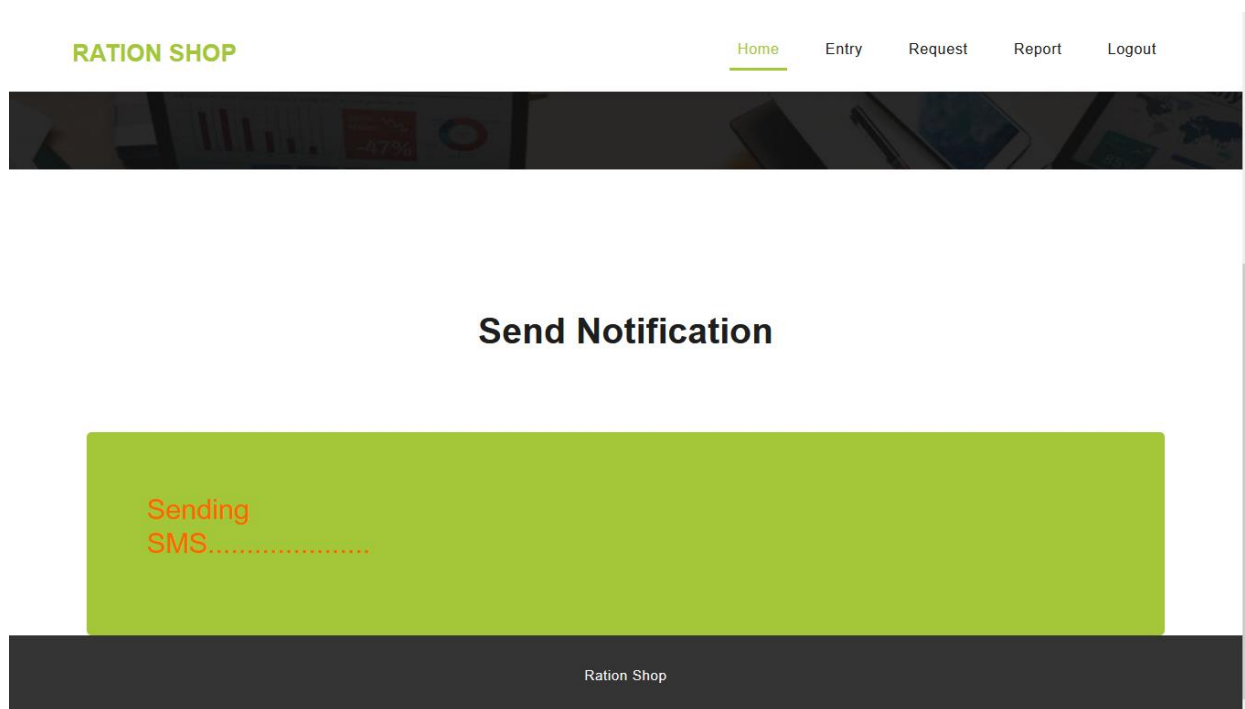


Figure 8.2.21 Send notification

RATION SHOP
Home
Entry
Request
Report
Logout

Time Slot: 11:00 to 11:10
Customer: 246302, Sathish [Deliver](#)
Customer: 246303, Murugan [Deliver](#)

Bill - Card: 246302 - Enter the Quantity

Rice - 2480 Kg

Wheat - 198 Kg

Sugar - 198 Kg

Oil - 99 Kg

SUBMIT

Time Slots:

S.No	Time	View
1	11:00 to 13:00	View

Figure 8.2.22 Time slot

RATION SHOP
Home
Entry
Request
Report
Logout

SUBMIT

Time Slots:

S.No	Time	View
1	11:00 to 13:00	View

1. Card No.: 246302
Time: 11:00 to 11:10
Delivered on 11:04
Rice-20,Wheat-2,Oil-1
Rs. 40

2. Card No.: 246303
Time: 11:00 to 11:10
Delivered on 11:08
Rice-20,Wheat-2,Sugar-2,Oil-1
Rs. 100

3. Card No.: 246304
Time: 11:10 to 11:20
Not Delivered

4. Card No.: 246305
Time: 11:10 to 11:20
Not Delivered

5. Card No.: 246306
Time: 11:20 to 11:30
Not Delivered

6. Card No.: 246307
Time: 11:20 to 11:30
Not Delivered

7. Card No.: 246308
Time: 11:30 to 11:40
Not Delivered

8. Card No.: 246309
Time: 11:30 to 11:40
Not Delivered

9. Card No.: 246310
Time: 11:40 to 11:50
Not Delivered

10. Card No.: 246311
Time: 11:40 to 11:50
Not Delivered

11. Card No.: 246312
Time: 11:50 to 12:00
Not Delivered

12. Card No.: 246313
Time: 11:50 to 12:00
Not Delivered

13. Card No.: 246314
Time: 12:00 to 12:10

14. Card No.: 246315
Time: 12:00 to 12:10

15. Card No.: 246316
Time: 12:10 to 12:20

Figure 8.2.23 List of Time slots

RATION SHOP
Home
Entry
Request
Report
Logout

Ration Entry: 02-01-2022

Time Slot: 11:10 to 11:20
Customer: 246304, Dinesh [Deliver](#)
Customer: 246305, Raj [Deliver](#)

Bill - Card: 246304 - Enter the Quantity
Rice - 2460 Kg
Wheat - 196 Kg
Sugar - 198 Kg
Oil - 98 Kg

Time Slots:

S.No	Time	View
1	11:00 to 13:00	View

1. Card No.: 246302 Time: 11:00 to 11:10	2. Card No.: 246303 Time: 11:00 to 11:10	3. Card No.: 246304 Time: 11:10 to 11:20
---	---	---

Figure 8.2.24 Enter the product quantity

RATION SHOP
Home
Entry
Request
Report
Logout

Shop Admin
RATION STOCKS

Available Stocks
[Stock Required](#)

S.No	Product	Available	Max. Deliver	Date
1	Rice	2440 Kg	20	30-12-2021
2	Wheat	196 Kg	2	02-01-2022
3	Sugar	196 Kg	2	02-01-2022
4	Oil	98 Kg	1	02-01-2022

Ration Shop

Figure 8.2.25 Shop admin view the available stocks

RATION SHOP
Home
Entry
Request
Report
Logout

Stocks Required

Required Product

Quantity

SUBMIT

Product

0

S.No	Product	Quantity	Date
1	Rice	100	03-01-2022

Ration Shop

Figure 8.2.26 Stocks Required

RATION SHOP
Home
Entry
Request
Report
Logout

Report

Date

dd / mm / yyyy

Month & Year

1

2022

Submit

S.No	Date	Time	Duration	Delivered
1	03-01-2022	11:00 - 13:00	10 Minutes	View

Figure 8.2.27 Report

RATION SHOP

Home

Entry

Request

Report

Logout

Number of Cards: 22

S.No	Card	Alloted Time	Deliver Status
1	246302	11:00 to 11:10	Rice-20,Wheat-2,Oil-1, Rs.40
2	246303	11:00 to 11:10	Rice-20,Wheat-2,Sugar-2,Oil-1, Rs.100
3	246304	11:10 to 11:20	Rice-20,Sugar-2, Rs.60
4	246305	11:10 to 11:20	Not Delivered
5	246306	11:20 to 11:30	Not Delivered
6	246307	11:20 to 11:30	Not Delivered
7	246308	11:30 to 11:40	Not Delivered
8	246309	11:30 to 11:40	Not Delivered
9	246310	11:40 to 11:50	Not Delivered
10	246311	11:40 to 11:50	Not Delivered
11	246312	11:50 to 12:00	Not Delivered
12	246313	11:50 to 12:00	Not Delivered
13	246314	12:00 to 12:10	Not Delivered
14	246315	12:00 to 12:10	Not Delivered
15	246316	12:10 to 12:20	Not Delivered

Figure 8.2.28 number of card details report

CHAPTER 9

REFERENCES

1. A. Mubarakh, M. I. Wahyuddin and S. Ningsih, "Queuing System Design On Android-Based Bank Teller Method Using Multi Channel - Single Phase", vol. 3, no. 4, 2020.
2. N. Andriyanov and V. Sonin, "The use of random process models and machine learning to analyze the operation of a taxi order service", ITM Web Conf, vol. 30, pp. 04014, 2019.
3. J. Chen, C. Du, P. Han, and X. Du, "Work-in-progress: non-preemptive scheduling of periodic tasks with data dependency upon heterogeneous multiprocessor platforms," in Proc. IEEE 40th Real-Time Syst. Symp. (RTSS), Dec. 2019, pp. 540–543, doi: 10.1109/RTSS46320.2019.00059.
4. J. Chen, C. Du, F. Xie, and B. Lin, "Scheduling non-preemptive tasks with strict periods in multi-core real-time systems," J. Syst. Archit., vol. 90, pp. 72–84, Oct. 2018, doi: 10.1016/j.sysarc.2018.09.002.
5. W. Bouazza, Y. Sallez, and B. Beldjilali, "A distributed approach solving partially flexible job-shop scheduling problem with a Q-learning effect," IFAC-PapersOnLine, vol. 50, no. 1, pp. 15890–15895, Jul. 2017.
6. Y.-R. Shiue, K.-C. Lee, and C.-T. Su, "Real-time scheduling for a smart factory using a reinforcement learning approach," Comput. Ind. Eng., vol. 125, pp. 604–614, Nov. 2018.
7. J. Shahrabi, M. A. Adibi, and M. Mahootchi, "A reinforcement learning approach to parameter estimation in dynamic job shop scheduling," Comput. Ind. Eng., vol. 110, pp. 75–82, Aug. 2017, doi: 10.1016/j.cie.2017.05.026.
8. Y.-F. Wang, "Adaptive job shop scheduling strategy based on weighted Q-learning algorithm," J. Intell. Manuf., vol. 31, no. 2, pp. 417–432, Feb. 2020, doi: 10.1007/s10845-018-1454-3.
9. C. Mogilner, H. E. Hershfield, and J. Aaker, "Rethinking time: Implications for well-being," Consumer Psychology Review, vol. 1, no. 1, pp. 41–53, 2018.
10. S. U" lku", C. Hydock, and S. Cui, "Making the wait worthwhile: Experiments on the effect of queueing on consumption," Management Science, 2019.