Logo

Description automatically generated

UDHAARI BOOK

by

Anis Tamboli (M.Sc. CA)

Sruthi Sunil (M.Sc. IMCA)

Suyasha Vilas Pujari (MCA)

Presented to the Faculty of

<Name of University>

In Partial Fulfillment

Of the Requirements

For the Degree

<NAME OF DEGREE>

THE UNIVERSITY OF <UNIV Name>

<MONTH OF GRADUATION, YEAR>

UDHAARI BOOK

by

Anis Tamboli

Sruthi Sunil

Suyasha Vilas Pujari

APPROVED BY

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

<Chair’s Name, Degree>, Chair

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

<Member’s Name, Degree >, Committee Member

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

<Member’s Name, Degree >, Committee Member

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

<Member’s Name, Degree >, Committee Member

RECEIVED/APPROVED BY THE COLLEGE OF <COLLEGE NAME>:

<Associate Dean’s Name, Degree>, Associate Dean

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

<Dean’s Name, Degree>, Dean

**Acknowledgements**

I would like to express my gratitude towards my parents, my college guides and all the team members of Bluepineapple for their kind co-operation and encouragement which helped me in completion of this project.

I am highly incredited to Bluepineapple Organization for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. I would like to show my greatest appreciation to SIR Suresh Burde (Principal Architect At Bluepineapple) for his immense support, encouragement & belief on the team.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities. I would like to extend my sincere thanks to all of them.

ABSTRACT

UDHAARI BOOK

Anis Tamboli

Sruthi Sunil

Suyasha Vilas Pujari

<University of Name of Univ, <year>

< Project> Chair: <Chair’s Name>

Co-Chair: <If applicable. Co-Chair’s Name>

Udhaari Book is a mobile application providing a platform for users namely, vendor (shopkeeper) and consumer to sell and buy products on credit. These products are sold to consumers who promise to pay the cost later mostly at the end of every month. The purchase and payment records of which are stored, saved and available to both the parties as and when required.

**TABLE OF CONTENTS**

Introduction 1

About the project 1

Application Features 2

Assumptions 2

Out-of-scope 2

Caveats 3

Feasibility study 3

Technical feasibility 4

Market research and analysis 4

Unique preposition & value creation 5

Stakeholder Analysis 5

Identification of stakeholders 5

Persona identification 6

Stakeholder engagement 6

Stakeholder communication 7

Project Plan 8

Project Management Plan 8

Project Scope 8

Risk Management plan 8

Identification, analysis and mitigation plan 9

Risk response Strategies 9

Project Team structure, roles and responsibilities 10

Project Tracking, Scheduling 10

Product Backlog 15

Scrum cycle, methodology 15

Requirement Analysis 16

Requirement collection methods used 16

Requirements finalisation process 16

Requirements documentation 16

User Stories 20

Detailed User stories description 23

Requirements Estimation 58

Effort Estimation Report 65

Design 67

Software Development Methodology 67

Peer review process 67

Collaboration Methodology 68

Coding Guidelines 68

Source Code Repository 70

Technical stack details 70

Technologies used, their versions 71

Database Design 73

ER Diagrams 73

Entity Description 76

Work Flow Diagram 81

Component Diagram 82

Sequence diagrams 83

Wireframes 86

Development 105

Application flow 105

Application features 107

Launching Screen 108

Vendor (shopkeeper) Login 109

Vendor (shopkeeper) Register 110

Add New Consumer 113

Vendor (shopkeeper) Account Details 117

Add Products 121

View All Products 124

Add new Product 126

Udhaari Records 127

Purchase 131

Payment 132

Make Payment 133

Consumer Login 136

Consumer Register 138

Consumer Dashboard 140

My Udhaari 141

Purchase 144

Payment 145

Consumer Account Details 146

Testing 147

Quality process 147

Test Cases identification 147

Test Cases Execution 190

Defect Tracking and analysis 217

Deployment & Implementation 218

Limitations & Future Enhancements 219

Limitations 219

Future Enhancement 219

Conclusion 221

References 222

Glossary: 223

List of Tables

Table 1 Risk Identification 9

Table 2.1 Categorized Features 17

Table 2.2 List Of Must Have User Stories 20

Table 3.1 User Story Description & Tasks - Launching Page 23

Table 3.2 Vendor Registration 24

Table 3.3 Vendor Login 26

Table 3.4 Vendor Dashboard 27

Table 3.5 Vendor Dashboard - Search Bar 28

Table 3.6 Add New Consumer 29

Table 3.7 Vendor – Navigation Tab 31

Table 3.8 Account Details 32

Table 3.9 Delete A Consumer Account 34

Table 3.10 View Products 35

Table 3.11 Add A New Product 36

Table 3.12 Add Udhaari/Products 37

Table 3.13 Total Amount/Current Bill 39

Table 3.14 Udhaari Records - Transaction History 40

Table 3.15 Purchase Transactions 41

Table 3.16 Payment Transactions 42

Table 3.17 Make Payment 43

Table 3.18 Back To Vendor Dashboard 45

Table 3.19 Vendor Logout 46

Table 3.20 Consumer Registration 47

Table 3.21 Consumer Login 48

Table 3.22 Consumer Dashboard 49

Table 3.23 Consumer Dashboard – Search Bar 50

Table 3.24 Consumer – Navigation Tab 51

Table 3.25 My Udhaari – Transaction History 52

Table 3.26 Purchase Transaction 53

Table 3.27 Payment Transaction 54

Table 3.28 My Account 55

Table 3.29 Back To Consumer Dashboard 56

Table 3.30 Consumer Logout 57

Table 3.31 Development Estimation 57

Table 4.1 Requirement Estimation 58

Table 4.2 Estimating Document Estimation 59

Table 4.3 Design Estimation 60

Table 4.4 Planning Estimation 61

Table 4.5 Study & Setup Estimation 62

Table 4.6 Testing Estimation 63

Table 4.7 Documentation Estimation 64

Table 4.8 Comparison Report Of All SDLC Phase Estimation 65

Table 4.9 Total Estimation Report 66

Table 4.10 Summary Of Estimation 66

Table 5.1 Technology Stack 70

Table 5.2 Technology, Libraries & Versions 71

Table 6.1 Database Tables - vendor 76

Table 6.2 consumer 76

Table 6.3 vendor\_consumer 77

Table 6.4 product 78

Table 6.5 consumer\_product\_vendor 78

Table 6.6 payment\_history 79

Table 6.7 transaction\_history 80

Table 7.1 UI Testing – Launching Page 148

Table 7.2 Vendor Registration 149

Table 7.3 Vendor Login 151

Table 7.4 Vendor Dashboard 153

Table 7.5 Vendor Dashboard – Search Bar 155

Table 7.6 Add New Consumer 156

Table 7.7 Vendor – Navigation Tab 158

Table 7.8 Edit Account 160

Table 7.9 Delete Account 162

Table 7.10 Add Products Table 163

Table 7.11 Total Amount/ Current Bill 165

Table 7.12 View Products 166

Table 7.13 Add A New Product 167

Table 7.14 Udhaari Records – Transaction History 168

Table 7.15 Purchase Transaction 170

Table 7.16 Payment Transaction 172

Table 7.17 Make Payment 173

Table 7.18 Back To Vendor Dashboard 175

Table 7.19 Vendor Logout 176

Table 7.20 Consumer Registration 177

Table 7.21 Consumer Login 179

Table 7.22 Consumer Dashboard 181

Table 7.23 Consumer Dashboard – Search Bar 183

Table 7.24 Consumer – Navigation Tab 184

Table 7.25 My Udhaari – Purchase Transaction 185

Table 7.26 Payment Transaction 186

Table 7.27 My Account 187

Table 7.28 Back To Consumer Dashboard 188

Table 7.29 Consumer Logout 189

Table 8.1 Backend Testing – Vendor Registration 190

Table 8.2 Vendor Login 191

Table 8.3 Vendor Dashboard 192

Table 8.4 Add New Consumer 193

Table 8.5 Vendor Dashboard – Search Bar 194

Table 8.6 Account Details 195

Table 8.7 Edit Account 196

Table 8.8 Delete Account 197

Table 8.9 Add Product 198

Table 8.10 Total Amount/Current Bill 200

Table 8.11 View Products 201

Table 8.12 Add A New Product 202

Table 8.13 Udhaari Records 203

Table 8.14 Transaction History 204

Table 8.15 Purchase Transaction 205

Table 8.16 Purchase History 206

Table 8.17 Payment Transaction 207

Table 8.18 Payment History 208

Table 8.19 View Threshold 209

Table 8.20 Make Payment 210

Table 8.21 Change Payment Details 211

Table 8.22 Update Payment Details 212

Table 8.23 Consumer Registration 213

Table 8.24 Consumer Login 214

Table 8.25 Consumer Dashboard 215

Table 8.26 My Account 216

Table 9 Defects, Cause & Resolution 217

Table 10 Glossary 223

List of Figures

Figure 1 Gantt Chart 12

Figure 2 Er Diagram 75

Figure 3 Workflow Diagram 81

Figure 4 Component Diagram 82

Figure 5.1 Vendor Sequence Diagram 84

Figure 5.2 Consumer Sequence Diagram 85

Figure 6.1 Wireframes - Launching Screen 86

Figure 6.2 Vendor Registration 87

Figure 6.3 Vendor Login 88

Figure 6.4 Vendor Dashboard 89

Figure 6.5 Add New Consumer 90

Figure 6.6 Account Details 91

Figure 6.7 Add Products 92

Figure 6.8 View Products 93

Figure 6.9 Transaction History 94

Figure 6.10 Payment Transaction 95

Figure 6.11 Purchase Transaction 96

Figure 6.12 Make Payment 97

Figure 6.13 Consumer Registration 98

Figure 6.14 Consumer Login 99

Figure 6.15 Consumer Dashboard 100

Figure 6.16 Transaction History 101

Figure 6.17 Payment Transaction 102

Figure 6.18 Purchase Transaction 103

Figure 6.19 My Account 104

Figure 7.1 Screen Shots – Launching Page 108

Figure 7.2 Vendor Login 109

Figure 7.3 Vendor Registration 110

Figure 7.4 Vendor Registration Successful 111

Figure 7.5 Vendor Dashboard 112

Figure 7.6 Add New Consumer 113

Figure 7.7 Search To Add A Registered Consumer 114

Figure 7.8 Set Account Start Date 115

Figure 7.9 Consumer Added Successfully 116

Figure 7.10 Account Details Of Selected Consumer 117

Figure 7.11 Edit Account 118

Figure 7.12 Delete Account 119

Figure 7.13 Account Deleted Successfully 120

Figure 7.14 Add Products 121

Figure 7.15 Total Amount/Current Bill 122

Figure 7.16 Bill Added Successfully 123

Figure 7.17 View Products 124

Figure 7.18 Search A Product 125

Figure 7.19 Add A New Product 126

Figure 7.20 Product Added Successfully 127

Figure 7.21 Transaction History 128

Figure 7.22 Purchase Transaction 129

Figure 7.23 Payment Transaction 130

Figure 7.24 Purchase Bill 131

Figure 7.25 Payment Bill 132

Figure 7.26 Make Payment 133

Figure 7.27 Payment Details 134

Figure 7.28 Payment Successful 135

Figure 7.29 Consumer Login 136

Figure 7.30 After Login Successful 137

Figure 7.31 Consumer Registration 138

Figure 7.32 Consumer Registration Successful 139

Figure 7.33 Consumer Dashboard 140

Figure 7.34 My Udhaari 141

Figure 7.35 Purchase Transaction 142

Figure 7.36 Payment Transaction 143

Figure 7.37 Purchase Bill 144

Figure 7.38 Payment Bill 145

Figure 7.39 Consumer Account Details 146

Figure 8.1 Backend Testing - Vendor Registeration 190

Figure 8.2 Vendor Login 191

Figure 8.3 Vendor Dashboard 192

Figure 8.4 Add New Consumer 193

Figure 8.5 Search Bar For Vendor Dashboard 194

Figure 8.6 Account Details 195

Figure 8.7 Edit Account 196

Figure 8.9 Add Product 199

Figure 8.10 Total Amount/Current Bill 200

Figure 8.11 View Products 201

Figure 8.12 Add A New Product 202

Figure 8.13 Udhaari Records 203

Figure 8.14 Transaction History 204

Figure 8.15 Purchase Transaction 205

Figure 8.16 Purchase History 206

Figure 8.17 Payment Transaction 207

Figure 8.18 Payment History 208

Figure 8.19 View Threshold 209

Figure 8.20 Make Payment 210

Figure 8.21 Change Payment Details 211

Figure 8.22 Update Payment Details 212

Figure 8.23 Consumer Registration 213

Figure 8.24 Consumer Login 214

Figure 8.25 Consumer Dashboard 215

Figure 8.26 My Account 216

Introduction

“Ab Udhaari Le, Vishwas Se”, is the motto of the currently developed mobile application titled, “Udhaari Book”. As the title itself says it all, the idea of building this application is to create a platform enabling transparency between the two roles vendor (shopkeeper) and consumer, where consumer buys products on credit or udhaari from the vendor and promises to pay the cost later. Leading to the development of such a system for maintaining these records which will be avaiable and beneficial to both vendors as well as consumers at ease.

About the project

Though technology has been changing vastly, even today people from all walks of life prefer to pay bills for water supply, electricity etc., at the end of every month based on a detailed description of consumption done.

In a similar way, we often see people buying products from their local vendors and asking them to add it to their account promising to pay it later or at the end of the month. These vendors still manage these records with the use of a book and pen. And most of the time these records are not available to the consumers to cross verify if required.

The objective is to design a mobile based application that allows these vendors to manage all these records on a digital platform. Also, allowing consumers to check their purchase as well as payment records as and when required. Thus, generating transparency between both the parties.

The software developed is a mobile based application with two entites categorized as, vendor (shopkeeper) and consumer. The software is accessible only after successful registration of the user. All purchase as well as payment records are well maintained in the application and both the entites can view and verify these records by logging in with their valid credentials whenever required.

Application Features

This section lists the features of the application in brief as follows,

* Selection of role based category.
* Registration page available for multiple vendors and consumers .
* Login page through which users can access the application by providing their credentials.
* Different role based functionalities for both the vendor (shopkeeper) as well as consumer.
* Product and credit management system.

Assumptions

This part describes certain assumptions taken into consideration due to various constraints. They are as follows,

* Currently consumers can only opt for offline payment. Only the option to add and view payment details is provided. It is assumed that consumer can pay the amount by any way to the vendor (shopkeeper) but vendor is required to clear the bill manually by make payment option available in the application.
* Shops are not categorised based on the products sold.

Out-of-scope

Certain scenarios which are beyond the control of the application as well as the development team and is thoroughly based on the choice of the user of the application is described here,

* To give someone udhaari or not is the vendor's (shopkeeper's)choice.
* To take udhaari or not is the consumer’s choice.

Caveats

This part of the documentation describes notice, warning, or word of caution provided to an individual or entity before they take any action while using the application,

* Fair Use policy - This application is meant to be used by individuals as per fair use policy, that users are responsible for connecting with each other – vendor (shopkeeper) who sells products and consumers who buy them.
* The application is not responsible for any disputes arising out of the transactions done via the platform. It is up to the individuals to settle that amicably.
* The application does not guarantee specific arrangements to be made, incase of loss of money, life, property, etc. It is only a medium for people to sell products and maintain records which will be transparent to both vendors and consumers.

Feasibility study

It gives information about the preliminary investigation that helps to take decision about whether study of system should be feasible for development or not. The main objective of a feasibility study in this system is to acquire problem scope instead of solving the problem. Thus, the output of a feasibility study gave a clarity in the decision making of complete nature and scope of the proposed system.

Technical feasibility

This type of feasibility study helps in exploring other alternatives to avoid any kind of risks during the development and implementation phases. It analyzes and determines whether the solution can be supported by existing technology or not. Basic knowledge of coding, javascript, html, css used in React is one of the must requirement. Other requirements form the hardware and software technical feasibility which are achievable and are as follows,

* Visual Studio Code – platform used for coding and development
* ReactNative – platform for mobile application development
* PostgreSql – database used for development
* Expo Client – development environment
* GitHub repository

Market research and analysis

This section focuses on different personas, target audience, and customers information gathering which helps to determine the viability and success of this application service among these people.

As a part of market analysis, following are the examinations done to understand the existing system, its features, pros and cons. Thus, giving a better understanding of how to go further with the development process from a bussiness perspective as well. Thus, leading to increase the demand and value of the product.

* Existing System – Many applications maintaining credit records are available in the market, but none of them have role specific functionalities.
* Need For System – The current application has role specific functionalities and is created majorly to be a transparent platform for the two roles vendor (shopkeeper) and consumer in specific.

Unique preposition & value creation

The existing system included features and functionalities which are not specific to roles, hence same features were implemented and available to all users irrespective of whether all these features were actually used by all users or not. Therefore, as a part of value proposition a new way of segmenting the market by introducing role specific features and different functionalities helps a novel value proposition to expand the market.

Stakeholder Analysis

This part describes information about all of the internal people and teams who this project involves or affects and are considered as [stakeholders](https://www.productplan.com/glossary/stakeholder/). They determine how best to involve and communicate each of these stakeholder groups throughout.

Identification of stakeholders

The following stakeholders are identified and grouped based on their levels of participation, interest, and influence in the project.

* Team leads
* Team members
* External teams
* College Guides/Professors
* Vendors (Shopkeepers)
* Consumers

Persona identification

The following personas are identified as the two roles around which the application revolves,

* Vendors – selling various kinds of products.
* Consumers - families with monthly expense planner, bachelors/spinsters, working employees.

Stakeholder engagement

As a part of stakeholder engagement, people whose decisions have direct or indirect impact and influence on overall project are identified and categorised as follows,

* Team leads : They support or oppose the decisions made for the overall development of the project, by giving their inputs in the form of suggestions and feedbacks. A thorough guidance throughout the project is always ensured from their side.
* Team members : All the Software Development Life Cycle (SDLC) phases are well discussed and planned in the initial stages of the project itself among the team members. Professional co-operation and co-ordination led to successful output of the application.
* External teams : Suggestions, critics, reviews and feedbacks from external team enables to look at the application from a different perspective leading to change certiain functionalities for easy access which are helpful for the target audience as well.
* College Guides/Professors : Discussions from guides help to improve the both UI as well as other functionalities of the application.
* Vendors & Consumers: In this case they are not aware of the system currently. But once they start using the system, they will be people who will be impacted by it.

Stakeholder communication

This part of the project, rightly refers to the regular exchange of information between the stakeholders for effective, progress of work. Weekly meetings and demos are well conducted with team leads and external team members so as to receive feedbacks and suggestions well in advance. Communication between team members is done on a regular base throughtout the week (5 days, 8 working hours) and twice a day so as to tackle runtime error scenarios and situations as a team. Thus, helping to understand the strengths and weakness of each other and continue to solve the problem together with unity.

Project Plan

This section covers the overall project plan from all possible areas of project, such as, time, cost, quality and scope. It also outlines the various stages and artifacts involved in the above areas with the strategies, chart, estimation tables which were followed for the entire Software Development Life Cycle (SDLC). The tool used for project planning is Gantt chart. They are used for scheduling and monitoring tasks, expenditure at all stages throughout the project, for communicating progress and producing reports. An estimation document was also made to give an approximate idea of how to continue with the project.

Project Management Plan

In this section, the factors taken into consideration which are [Scope](https://en.wikipedia.org/wiki/Scope_(project_management)) management, [Requirements management](https://en.wikipedia.org/wiki/Requirements_management), [Schedule](https://en.wikipedia.org/wiki/Schedule_(project_management)) management, [Quality management](https://en.wikipedia.org/wiki/Quality_management), [Resource management](https://en.wikipedia.org/wiki/Resource_management), [Stakeholder management](https://en.wikipedia.org/wiki/Stakeholder_management), [Communications management](https://en.wikipedia.org/wiki/Communications_management), Project [change management](https://en.wikipedia.org/wiki/Change_management), [Risk management](https://en.wikipedia.org/wiki/Risk_management).

Project Scope

The scope of the project decided is mostly based on time constraint as the time period to complete this project was approximately 60-65 days. Most of the focus was given on completing the must have essential requirements noted down. Every part of the project was equally distributed among resource and a strict plan enabled to achieve the successful development of this application.

Risk Management plan

It provides information about the plan to prepare to foresee risks, estimate impacts and define responses to risks. Here, a risk is "an uncertain event or condition that, if it occurs, has a positive or negative effect on the project's objectives”. A plan to control or avoid these risks form a very important aspect of the entire project so as to avoid any possibilitites or causes leading to the failure of the actual product.

Identification, analysis and mitigation plan

This section involves taking action to reduce exposure to potential risks and reduce the likelihood that those risks will happen again. Risk mitigation is one of the steps in risk management, which includes identifying the risk, analyzing the risk and mitigating the risk. The risk identified, analysed and mitigated are as follows,

Table 1 Risk Identification

|  |  |  |
| --- | --- | --- |
| **Risk Identification** | **Analysis** | **Mitigation Plan** |
| Allocate ownership of opportunity to a third-party | Security issues may arise | Provide authorities to third party |
| Runtime exception can occur | Warning may occur while project execution | Exception handling is performed on project |
| Database connectivity Issue | Database connectivity not possible if password is wrong | Blank password is set to database |

Risk response Strategies

This part convey information about the types of strategies applied to control risks identified and they are as follows,

* Remove the Risk - the best strategy is to remove the risk
* Reduce Impact - the next strategy is to try to reduce the impact of it, if it happens
* Reduce Likelihood of the risk occurring
* Transfer the Risk to something/someone else
* Contingency Plan - a plan to have an alternate arrangement
* Accept the Risk if nothing else is possible

Project Team structure, roles and responsibilities

The team involved for the development of this project consists of three members. The involvement and contribution of every member was equal throughout all the Software Development Life Cycle phases. Every member was assigned a particular number of user stories in the development phase which led to the completion of the development as expected.

Project Tracking, Scheduling

It involves, the pattern of tracking and scheduling the project plan. The tool used Gantt chart is **created using a Gannt Cemplate available in google add-ons.** The project is broken down into individual tasks which are listed in rows on the chart. Each task has a timeline extending out to the deadline of the task shown as a shaded block. These shaded time blocks are colour coded indicating tasks being delivered by team members.

Overlaying the time blocks is a progress line, showing how much work has been done on the task so far, and also the important milestones along the way. The milestones are typically annotated to indicate their relevance in percentages. Estimated and actual dates are also added at the end of the time blocks.

The screenshots of the Gantt chart representing the tasks assigned to each resource is as follows,

Table

Description automatically generated with medium confidenceFigure 1 Gantt Chart

Table

Description automatically generated

**Graphical user interface, text, application, table

Description automatically generated**

Product Backlog

Product backlog contains the decision-making artifact that helps estimate, refine, and prioritize everything that might sometime in the future be implemented. Some of these features for this product includes the online payment, a to-buy list for consumers, SMS notification on any important transaction or action being performed by any user.

Scrum cycle, methodology

[Scrum](https://university.hygger.io/en/articles/3036310-scrum-methodology) is a popular Agile framework designed for incremental product development. It has empirical nature and empowers teams to hypothesize regarding working patterns, test their ideas, process the experience, and perform necessary adjustments. Since it operates through iterations called Sprints, the main events and artifacts of a Sprint represent the components of a Scrum lifecycle. The time box which the team followed to deliver a specific list of agreed-upon items fitting into the confirmed definition of “done” is shown on the gantt chart above.The entire Scrum team equally engaged in the following responsibilities,

* Breaking down the requirements, creating tasks, estimating, and distributing them.
* Arranging Daily Scrum meetings in the morning and evening to keep track of the work being completed.
* Ensuring that product will be delivered at the required time .
* Updating statuses and the remaining efforts for their tasks to allow the creation of the final product.

Requirement Analysis

This part of the analysing describes, the process defining user expectations for a new software being built or modified commonly known as requirements gathering or requirements capturing. Thus, the project requirements analysis for this project encompasses those tasks that go into determining the needs or conditions to meet for the application being built taking account of all possible requirements from the client, the various stakeholders, analyzing, documenting, validating and managing software or system requirements.

Requirement collection methods used

Most of the requirements were directly gathered from the clients itself namely the vendors selling various products, the consumers from all age as well as income categories. Other requirements were collected through discussions with the team leads and team members. Information available on the web also formed a medium of source to identify and gather these requirements.

Requirements finalisation process

Requirements were first categorized as “must have” and “nice to have” requirements wherein, the must have requirements included the ones that could fulfill the basic needs of this application which during requirement analysis was also found to be mentioned by most of the sources. The nice to have requirements were short listed and best among them were finalised for the development if and as time permits.

Requirements documentation

As this application revolves around role based requirements, each role has been given the following features and functionalities,

Table 2.1 Categorized Features

|  |  |  |
| --- | --- | --- |
| **Roles** | **Must have features** | **Nice to have features** |
| Vendor (shopkeeper) | 1. Launching page – to select category Vendor (shopkeeper) 2. Registration page – to register with following fields,  * Name * Contact Number * Shop Name * Shop Address * Password  1. Login page – to login with valid credentials for the following fields,  * Registered Mobile Number (RMN) * Password  1. Dashboard – consisting of the following functionalities,  * Search bar to search consumers using their name or RMN * Floating button to add a new consumer * Logout button * List of all consumers, with their RMN and outstanding amount to be paid onclick of which opens a tab navbar with different functionalities available for each consumer  1. Tab navbar - consisting the following navigation options  * Account Details which gives details such as address, account status, name, RMN, threshold, account start date, billing start date, due date, last paid amount, total due amount, partial due amount, option to edit and save changes and a delete account button for the selected consumer * Add Products which includes view all products, add new product, add bill by entering product and quantity for which base price, total price is auto calculated with total amount also being calculated dynamically as the clear bill button * Udhaari Records which includes all transactions, in specific purchase and payment transactions in a bill format * Make Payment which includes details of total and partial due amount to be paid with remaining amount changing depending on the amount being paid * Home button to go back to the Dashboard * Logout button | 1. Scan and detect the products 2. SMS notification confirming that consumer has been successfully added to a vendors account at the time of adding a new consumer 3. Language preference |
| Consumer | 1. Launching page – to select category Consumer 2. Registration page – to register with following fields,  * Name * Contact Number * Address * Password  1. Login page – to login with valid credentials for the following fields,  * Registered Mobile Number (RMN) * Password  1. Dashboard – consisting of the following functionalities,  * Search bar to search vendor (shopkeeper) using their shop name or RMN * Logout button * List of all vendor (shopkeeper)’s shop name, with their RMN and outstanding amount to be paid onclick of which opens a tab navbar with two functionalities available for each vendor (shopkeeper)  1. Tab navbar - consisting the following navigation options  * My Udhaari which includes all transactions, in specific purchase and payment transactions in a bill format * Account Details which gives details such as shop address, account status, consumer name, RMN, threshold, account start date, billing start date, due date, last paid amount, total due amount and partial due amount * Home button to go back to the Dashboard * Logout button | 1. Make Payment (Online) |

User Stories

This part of the project describes, user stories which is a tool used in Agile software development to capture a description of a software feature from an end-user perspective. It describes the type of user, what they want and why. A user story helps to create a simplified description of a requirement. Here is the list of all user stories included in the software development,

Table 2.2 List Of Must Have User Stories

|  |  |
| --- | --- |
| **Sr. No.** | **User Stories** |
|  | As a user, I want to, select a role so that I can login |
|  | As a new unregistered vendor (shopkeeper), I want to, register so that I can login |
|  | As a vendor (shopkeeper), I want to, see a login form so that I can enter my credentials to access the Dashboard |
|  | As a vendor (shopkeeper), I want to, see   * the list of all the consumers * search bar to search a consumer * floating button to add a new consumer * logout button   on the Dashboard screen |
|  | As a vendor (shopkeeper), I want to, search for a particular consumer using name or RMN |
|  | As a vendor (shopkeeper), I want to, see a form to add a new consumer |
|  | As a vendor (shopkeeper), I want to, see different options to navigate after selecting a consumer on the screen |
|  | As a vendor (shopkeeper), I want to, see and edit the account details with the total amount to be paid of a particular consumer on the screen |
|  | As a vendor (shopkeeper), I want to, delete a selected consumer’s account |
|  | As a vendor (shopkeeper), I want to, view all products available with their base price |
|  | As a vendor (shopkeeper), I want to, add a new product to the product list with its base price |
|  | As a vendor (shopkeeper), I want to, add purchased products for a selected consumer |
|  | As a vendor (shopkeeper), I want to, see the total amount changing dynamically |
|  | As a vendor (shopkeeper), I want to, see all transaction history on the screen |
|  | As a vendor (shopkeeper), I want to, see only the purchase transactions with total amount to be paid in a bill format on the screen |
|  | As a vendor (shopkeeper), I want to, see only the payment transactions with total amount paid in a bill format on the screen |
|  | As a vendor (shopkeeper), I want to, make payment for a selected consumer |
|  | As a vendor (shopkeeper), I want to, go back to the Dashboard of the application |
|  | As a vendor (shopkeeper), I want to, logout of the application |
|  | As a new unregistered consumer, I want to, register so that I can login |
|  | As a consumer, I want to, see a login form so that I can enter my credentials to access the Dashboard |
|  | As a consumer, I want to, see   * the list of all my vendors * search bar to search a vendor (shopkeeper) * logout button   on the Dashboard screen |
|  | As a consumer, I want to, search for a particular vendor (shopkeeper) using shop name or RMN |
|  | As a consumer, I want to, see different options to navigate after selecting a vendor (shopkeeper) on the screen |
|  | As a consumer, I want to, see all transaction history on the screen |
|  | As a consumer, I want to, see only the purchase transactions with total amount to be paid in a bill format on the screen |
|  | As a consumer, I want to, see only the payment transactions with total amount paid in a bill format on the screen |
|  | As a consumer, I want to, see all my account details for selected vendor (shopkeeper) on the screen |
|  | As a consumer, I want to, go back to the Dashboard of the application |
|  | As a consumer, I want to, logout of the application |

Detailed User stories description

This part of the project describes, detailed description of each user story mentioned above with the tasks listed to complete its development along with the estimated as well as actual time required to achieve it.

Here 1 unit = 30 mins on which estimated and actual time is calculated

Table 3.1 User Story Description & Tasks - Launching Page

|  |  |
| --- | --- |
| **Sr. No.** | 1 |
| **User Story** | As a user, I want to, select role so that I can login |
| **Description** | Given I’m a user and want to select a login category  When I open the Udhaari Book application  Then I want to see two categories named vendor (shopkeeper) and consumer to login as per my role |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a Launching component with application name and dynamic image | 6 | 4 |
|  | Routing using Navigation to vendor (shopkeeper) Login page | 4 | 3 |
|  | Routing using Navigation to consumer Login page | 4 | 3 |
| **TOTAL** |  | **14** | **10** |

Table 3.2 Vendor Registration

|  |  |
| --- | --- |
| **Sr. No.** | 2 |
| **User Story** | As a new unregistered vendor (shopkeeper), I want to, register so that I can login |
| **Description** | Given I’m a non-registered vendor (shopkeeper) and I’m on the ‘Registration’ page  When I fill the   * Contact Number * Name * Shop Name * Shop Address * Password   fields with my credentials and I click the register button  Then the application registers me and opens the ‘Login’ page |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create Registration component with fields contact number, password, name, shop name and address | 8 | 4 |
|  | Write methods for validating user inputs, and show error message or success message depending on the result in the front end | 8 | 9 |
|  | Write onPress() handler giving notification on successful registration | 4 | 2 |
|  | Write backend validation methods for database crud operations to store and access the details | 8 | 10 |
|  | Write exception handling methods for backend and frontend | 6 | 1 |
|  | Routing using Navigation to Vendor (shopkeeper) Registration page | 4 | 1 |
| **TOTAL** |  | **38** | **27** |

Table 3.3 Vendor Login

|  |  |
| --- | --- |
| **Sr. No.** | 3 |
| **User Story** | As a vendor (shopkeeper), I want to, see a login form so that I can enter my credentials to access the Dashboard |
| **Description** | Given I’m a registered vendor (shopkeeper) and I’m on the ‘Login’ page  When I select vendor login and fill the   * Registered Mobile Number (RMN) * Password   fields with my credentials and I click the login button  Then the application signs me in and redirects to the ‘Dashboard’ |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create Login component with fields contact number and password | 8 | 4 |
|  | Write methods for validating user inputs, and show error message or success message depending on the result in the front end | 6 | 6 |
|  | Create authentication APIs and write backend validation methods for database crud operations to access the details | 12 | 8 |
|  | Routing using Navigation to Dashboard | 4 | 2 |
| **TOTAL** |  | **30** | **20** |

Table 3.4 Vendor Dashboard

|  |  |
| --- | --- |
| **Sr. No.** | 4 |
| **User Story** | As a vendor (shopkeeper), I want to, see   * the list of all the consumers * search bar to search a consumer * floating button to add a new consumer * logout button   on the Dashboard screen |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper)  When I’m on the ‘Dashboard’  Then I want to see the following options   * all the consumers with their name and RMN as contact list so that I can select any consumer * a search bar to search my registered consumer by name or RMN so that I can select that specific consumer * a floating button to add new consumers and start their Udhaari * a logout button to logout from my account |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a Dashboard with search bar on the top, logout button, add floating button | 8 | 10 |
|  | UI of consumers list | 6 | 8 |
|  | Write backend validation methods for database fetch operation to access the details | 10 | 10 |
| **TOTAL** |  | **24** | **28** |

Table 3.5 Vendor Dashboard - Search Bar

|  |  |
| --- | --- |
| **Sr. No.** | 5 |
| **User Story** | As a vendor (shopkeeper), I want to, search for a particular consumer using name or RMN |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on the ‘Dashboard’  When I click the search bar and fill the consumer’s   * RMN or * Name   and click the search button  Then the application displays that consumer with searched RMN or consumers with searched name respectively |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Write onPress() event handler to search for consumer | 5 | 4 |
|  | Write methods for validating user inputs, and show error message or success message depending on the search result in the front end | 6 | 6 |
|  | Write backend validation methods for database crud operations to store and access the details | 10 | 8 |
|  | Write exception Handling for all frontend and backend | 5 | 2 |
|  | Routing using Navigation to Account Details page | 4 | 10 |
| **TOTAL** |  | **30** | **30** |

Table 3.6 Add New Consumer

|  |  |
| --- | --- |
| **Sr. No.** | 6 |
| **User Story** | As a vendor (shopkeeper), I want to, see a form to add a new consumer |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on the ‘Dashboard’  When I click the ‘Add Floating button’ and search the registered consumer’s RMN  Then it auto generates search for name and address of that consumer after that  When I fill the   * threshold * start date * due date   fields and click on add consumer button  Then the application adds the new consumer to consumer’s list |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Add consumer form with fields contact number, threshold, start date and due date | 6 | 8 |
|  | Write methods for validating user inputs, and show error message or success message depending on the result in the front end | 6 | 8 |
|  | Write backend validation methods for database crud operations to store and access the details | 8 | 10 |
|  | Calendar date picker for dates selection | 8 | 8 |
|  | Write backend methods using date fns to perform date operations and store them in the database | 6 | 4 |
|  | Routing using Navigation to add consumer page and vice versa with onPress() event handler to open the form for a new consumer | 10 | 6 |
| **TOTAL** |  | **44** | **44** |

Table 3.7 Vendor – Navigation Tab

|  |  |
| --- | --- |
| **Sr. No.** | 7 |
| **User Story** | As a vendor (shopkeeper), I want to, see different options to navigate after selecting a consumer on the screen |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on the ‘Dashboard’  When I select a particular consumer  Then the application gives a navbar that shows the following options: account details, add products, udhaari records, make payment, logout and back to Dashboard so that I can perform these tasks specific to that consumer |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a navbar including the following labels Account Details, Add Products, Udhaari Records, Make Payment and Logout as the navbar options | 6 | 6 |
|  | Create Account Details component for displaying the selected consumer’s details | 6 | 6 |
|  | Create display label for showing active status | 6 | 2 |
|  | Create a back button | 2 | 1 |
|  | Routing using Navigation to Account Details with on Press(),  Add Products, Udhaari Records, Make Payment, Back to Dashboard and Logout | 14 | 7 |
| **TOTAL** |  | **34** | **22** |

Table 3.8 Account Details

|  |  |
| --- | --- |
| **Sr. No.** | 8 |
| **User Story** | As a vendor (shopkeeper), I want to, see and edit the account details with the total amount to be paid of a particular consumer on the screen |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and  When I select ‘Account Details’  Then the application gives the name and RMN on top part of the screen  The application also shows a status option at the upper right corner showing whether the account of that consumer is active or blocked.  Other details like starting date of account, threshold, last paid amount, current start date, current due date, total due amount, partial due amount is also displayed on the screen along with an inline edit option  When I click the editable fields  Then the application allows me to edit the name and threshold field (0-1) and save it and also shows all account details where the old fields are replaced with the new ones |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create an edit button to edit consumer name and threshold in Account Details page | 8 | 4 |
|  | Create an API to fetch  data such as name, RMN, account started date, threshold, last paid amount, start date, due date, total due amount and partial due amount | 12 | 18 |
|  | Write methods for validating user inputs, and show error message or success message depending on the result in the front end | 6 | 6 |
|  | Create a save button to save changes and event onChange() handler | 4 | 4 |
|  | Write a post method for validating and updating the changed fields in the database from the backend | 6 | 8 |
| **TOTAL** |  | **36** | **40** |

Table 3.9 Delete A Consumer Account

|  |  |
| --- | --- |
| **Sr. No.** | 9 |
| **User Story** | As a vendor (shopkeeper), I want to, delete a selected consumer’s account |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on a selected consumer’s ‘Account Details’ page  When I click on the ‘Delete icon’  Then application deletes that consumer’s account and removes the name from the consumers list |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a delete icon on the Account Details page | 5 | 5 |
|  | Write event handler - onPress() give delete confirmation notification and based on user input delete the consumer’s name from the consumers list | 8 | 6 |
|  | Write a method to access data from the database in the backend | 6 | 6 |
|  | Routing using navigation to Dashboard | 4 | 1 |
| **TOTAL** |  | **23** | **18** |

Table 3.10 View Products

|  |  |
| --- | --- |
| **Sr. No.** | 10 |
| **User Story** | As a vendor (shopkeeper), I want to, view all products available with their base price |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on a selected consumer’s ‘Add Products’ page  When I click on the ‘View All Products’’  Then application displays all the available products with their base price |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a view all products page with search bar on the top, close button and floating button to add new product. | 6 | 5 |
|  | UI of products list | 4 | 3 |
|  | Write backend validation methods for database fetch operation to access the details | 4 | 4 |
|  | Create modal to add new product | 3 | 4 |
| **TOTAL** |  | **17** | **16** |

Table 3.11 Add A New Product

|  |  |
| --- | --- |
| **Sr. No.** | 11 |
| **User Story** | As a vendor (shopkeeper), I want to, add a new product to the product list with its base price |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on a selected consumer’s ‘Add Products’ page  When I click on the ‘+ New Product’  Then application displays a modal through which I can add a new product with its base price |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
| 1. | Create a modal with Product name and base price. | 4 | 3 |
| 2. | Create add and cancel button | 2 | 2 |
| 3. | Write backend validation methods for database fetch operation to access the details | 4 | 4 |
| 4. | Write exception Handling for all frontend and backend | 4 | 4 |
| **TOTAL** |  | **14** | **13** |

Table 3.12 Add Udhaari/Products

|  |  |
| --- | --- |
| **Sr. No.** | 12 |
| **User Story** | As a vendor (shopkeeper), I want to, add purchased products for a selected consumer |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I have selected a consumer  When I select add products option on the navbar  Then the application gives me a option to enter the product and quantity When I click add button  Then the application auto generates a row to the table containing the following columns   * product * quantity * base price * total price   where total price is auto calculated as quantity \* base price |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create Add Products component for adding purchase details | 4 | 4 |
|  | Create a table with the following fields DateTime, Product, Quantity, Base Price, Total Price and dynamic text showing Total Amount on adding a new product | 8 | 16 |
|  | Create an add button on clicking every new row | 8 | 12 |
|  | Write event handler - onPress() for this add button to validate and store these details | 5 | 12 |
|  | Write a validation method for backend processing | 6 | 3 |
|  | Write a post method for adding these fields in the database from the backend | 6 | 3 |
| **TOTAL** |  | **37** | **50** |

Table 3.13 Total Amount/Current Bill

|  |  |
| --- | --- |
| **Sr. No.** | 13 |
| **User Story** | As a vendor (shopkeeper), I want to, see the total amount changing dynamically |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on selected consumer’s ‘Add Products’ page  When I add a new product and click on ‘Add’ button  Then the application displays the total amount at the bottom part of the screen and auto calculates the total amount which is the sum of the total price of each added product  When I click on the add current bill button clears the table  Then the entry of that product is added to the purchase history |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Write a method to calculate multiplication of quantity and base price of the product to get total price | 4 | 2 |
|  | Write a method for adding all total prices to get the final total amount | 6 | 3 |
|  | Fetch total amount from database and display on add product page | 6 | 2 |
|  | Store updated total amount and total price in the database | 4 | 3 |
| **TOTAL** |  | **20** | **10** |

Table 3.14 Udhaari Records - Transaction History

|  |  |
| --- | --- |
| **Sr. No.** | 14 |
| **User Story** | As a vendor (shopkeeper), I want to, see all transaction history on the screen |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper)  When I click on selected consumer’s ‘Udhaari Records’ page  Then the application shows all transactions both payment and purchase history on the screen |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | onPress() of Udhaari Records create a component, which displays a list of all transactions, payment and purchase history toggle bars | 2 | 2 |
|  | Create filters for month and year | 2 | 2 |
|  | Create all APIs to fetch data such as all transactions, payment and purchase records | 1 | 1 |
|  | Routing using navigation after selecting purchase and payment bill page | 1 | 1 |
| **TOTAL** |  | **6** | **6** |

Table 3.15 Purchase Transactions

|  |  |
| --- | --- |
| **Sr. No.** | 15 |
| **User Story** | As a vendor (shopkeeper), I want to, see only the purchase transactions with total amount to be paid in a bill format on the screen |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on selected consumer’s ‘Udhaari Records’ page  When I click on purchase history of that consumer  Then the application shows a filter to see any years transaction, for easy access the application shows the bill amount, bill date, on clicking of it shows me the details of purchased product in a bill format |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create Udhaari Records component containing two radio buttons, purchase history and payment  history | 6 | 6 |
|  | Create a purchase history component which will display a list of products purchased by a selected consumer such as date, product, quantity, base price, total price of each product in tabular format. | 14 | 12 |
|  | Create filters for month and year | 8 | 8 |
|  | Create an  API to fetch and display the following details date, product, quantity, base price, total price from the database based on the filtered date | 10 | 8 |
|  | Routing using navigation to payment history | 4 | 1 |
| **TOTAL** |  | **42** | **35** |

Table 3.16 Payment Transactions

|  |  |
| --- | --- |
| **Sr. No.** | 16 |
| **User Story** | As a vendor (shopkeeper), I want to, see only the payment transactions with total amount paid in a bill format on the screen |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on selected consumer’s ‘Udhaari Records’ page  When I click on payment history of that consumer  Then the application shows a filter to see any years transaction, for easy access the application shows the paid amount, paid date along with the information of carry forwarded amount in bill format on clicking of it |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a payment history component, which displays a list of all previous payments paid by consumer | 8 | 8 |
|  | Create filters for year | 6 | 4 |
|  | Create an API to fetch data such as the date on which the consumer had made payment and the total amount paid by the consumer from the database | 10 | 6 |
|  | Routing using navigation to purchase history | 4 | 1 |
| **TOTAL** |  | **28** | **19** |

Table 3.17 Make Payment

|  |  |
| --- | --- |
| **Sr. No.** | 17 |
| **User Story** | As a vendor (shopkeeper), I want to, make payment for a selected consumer |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on selected consumer’s ‘Make Payment’ page  When I fill currently paying amount which is greater than the threshold and click on update balance  Then the application updates the remaining amount based on which the status is changed to full or partial payment and the bill is cleared |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a Make Payment component, which displays data such as total amount, paying date, currently paying amount and remaining amount. | 10 | 10 |
|  | Create an Update Balance button | 4 | 6 |
|  | Fetch values of total amount, paying date from database | 8 | 8 |
|  | Write a method to check if the currently paying amount is greater than the partial amount or not. | 4 | 4 |
|  | Display validation error if currently paying amount is less than partial amount. | 4 | 6 |
|  | Write a method to calculate the remaining amount by subtracting total amount and currently paying amount | 4 | 4 |
|  | Write an event handler onPress() which gives Payment Successful notification it should update the total amount and carried amount in the history | 8 | 10 |
| **TOTAL** |  | **42** | **48** |

Table 3.18 Back To Vendor Dashboard

|  |  |
| --- | --- |
| **Sr. No.** | 18 |
| **User Story** | As a vendor (shopkeeper), I want to, go back to the Dashboard of the application |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper)  When I click on Home button  Then the application redirects me to the Dashboard  page |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a home button on different pages such as Account Details, Add Products, Udhaari Records and Make Payment. | 4 | 8 |
|  | Route using navigation with onPress() event handler | 4 | 2 |
| **TOTAL** |  | **8** | **10** |

Table 3.19 Vendor Logout

|  |  |
| --- | --- |
| **Sr. No.** | 19 |
| **User Story** | As a vendor (shopkeeper), I want to, logout of the application |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper)  When I click on ‘Logout’ button  Then the application redirects me to the Login page |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a logout button on different pages such as Account Details, Add Products, Udhaari Records and Make Payment. | 4 | 8 |
|  | Route using navigation with onPress() event handler | 4 | 2 |
| **TOTAL** |  | **8** | **10** |

Table 3.20 Consumer Registration

|  |  |
| --- | --- |
| **Sr. No.** | 20 |
| **User Story** | As a new unregistered consumer, I want to, register so that I can login |
| **Description** | Given I’m a non-registered consumer and I’m on the ‘Registration’ page  When I fill the   * Name * Contact number * Address * Password   fields with my credentials and I click the ‘Register’ button  Then the application registers me and opens the Login page |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create Registration component with fields name, contact number, address and password | 4 | 2 |
|  | Write methods for validating user inputs, and show error message or success message depending on the result in the front end | 4 | 2 |
|  | Write onChange() handler giving notification on successful registration | 4 | 1 |
|  | Write backend validation methods for database crud operations to store and access the details | 6 | 2 |
|  | Write exception handling for all methods frontend and backend | 4 | 2 |
|  | Routing using Navigation to Consumer Login page | 4 | 1 |
| **TOTAL** |  | **26** | **10** |

Table 3.21 Consumer Login

|  |  |
| --- | --- |
| **Sr. No.** | 21 |
| **User Story** | As a consumer, I want to, see a login form so that I can enter my credentials to access the Dashboard |
| **Description** | Given I’m a registered consumer and I’m on the ‘Login’ page  When I select consumer login and fill the   * Registered Mobile Number (RMN) * Password   fields with my credentials and I click the ‘Login’ button  Then the application signs me in and redirects to the Dashboard |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create Login component with fields RMN and password | 4 | 4 |
|  | Write methods for validating user inputs, and show error message or success message depending on the result in the front end | 4 | 4 |
|  | Create authentication APIs and write backend validation methods for database crud operations to access the details | 8 | 6 |
|  | Routing using Navigation to the Consumer Dashboard | 4 | 1 |
| **TOTAL** |  | **20** | **15** |

Table 3.22 Consumer Dashboard

|  |  |
| --- | --- |
| **Sr. No.** | 22 |
| **User Story** | As a consumer, I want to, see   * the list of all my vendors * search bar to search a vendor (shopkeeper) * logout button   on the Dashboard screen |
| **Description** | Given I’m a successfully logged-in consumer  When I’m on the ‘Dashboard’  Then I want to see the following options   * list of all vendors and the total due amount to be paid to that vendor (shopkeeper). * a search bar to search my vendors by their shop name or RMN * a Logout button to logout from my account |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a Dashboard with search bar on the top, logout button | 4 | 2 |
|  | UI of vendors shop list | 4 | 3 |
|  | Write backend validation methods for database fetch operation to access the details | 6 | 6 |
|  | Write exception handling for all methods frontend and backend | 6 | 4 |
| **TOTAL** |  | **20** | **15** |

Table 3.23 Consumer Dashboard – Search Bar

|  |  |
| --- | --- |
| **Sr. No.** | 23 |
| **User Story** | As a consumer, I want to, search for a particular vendor (shopkeeper) using shop name or RMN |
| **Description** | Given I’m a successfully logged-in consumer and I’m on the ‘Dashboard’  When I click the search bar and fill the vendor (shopkeeper)’s   * Shop name or * RMN   and click the search button  Then the system displays that vendor with searched shop name, RMN and total amount to be paid |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Write onPress() event handler to search for vendor (shopkeeper) | 4 | 3 |
|  | Write methods for validating user inputs, and show error message or success message depending on the search result in the front end | 4 | 3 |
|  | Write backend validation methods for database crud operations to store and access the details | 6 | 6 |
|  | Write exception handling for all methods frontend and backend | 2 | 2 |
|  | Routing using Navigation to My Details with on Press() event handler to My Udhaari page | 3 | 2 |
| **TOTAL** |  | **19** | **16** |

Table 3.24 Consumer – Navigation Tab

|  |  |
| --- | --- |
| **Sr. No.** | 24 |
| **User Story** | As a consumer, I want to, see different options to navigate after selecting a vendor (shopkeeper) on the screen |
| **Description** | Given I’m a successfully logged-in consumer and I’m on the ‘Dashboard’  When I select a particular vendor’s (shopkeeper’s) shop  Then the application gives a navbar that shows the following options: My Udhaari, Account Details, Home and Logout so that I can perform these tasks specific to that vendor |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Write event handler - onChange() to change UI on selecting a particular consumer | 6 | 4 |
|  | Create a navbar including the following labels my udhaari, my account and logout | 4 | 2 |
|  | Create My Udhaari component for displaying the selected consumer’s details | 4 | 2 |
|  | Write exception handling for all methods frontend and backend | 6 | 3 |
|  | Routing using Navigation to My Udhaari, My account and Logout | 6 | 1 |
| **TOTAL** |  | **26** | **12** |

Table 3.25 My Udhaari – Transaction History

|  |  |
| --- | --- |
| **Sr. No.** | 25 |
| **User Story** | As a consumer, I want to, see all transaction history on the screen |
| **Description** | Given I’m a successfully logged-in consumer  When I click on selected consumer’s ‘My Udhaari’ page  Then the application shows all transactions both payment and purchase history on the screen |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | onPress() of Udhaari Records create a component, which displays a list of all transactions, payment and purchase history toggle bars | 2 | 2 |
|  | Create filters for month and year | 2 | 2 |
|  | Create all APIs to fetch data such as all transactions, payment and purchase records | 1 | 1 |
|  | Routing using navigation after selecting purchase and payment bill page | 1 | 1 |
| **TOTAL** |  | **6** | **6** |

Table 3.26 Purchase Transaction

|  |  |
| --- | --- |
| **Sr. No.** | 26 |
| **User Story** | As a consumer, I want to, see only the purchase transactions with total amount to be paid in a bill format on the screen |
| **Description** | Given I’m a successfully logged-in consumer and I’m on selected vendor (shopkeeper)’s ‘My Udhaari’ page  When I click on purchase history of that vendor (shopkeeper)  Then the application shows a filter to see any years transaction, for easy access the application shows the bill amount, bill date, on clicking of it shows me the details of purchased product in a bill format |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a navbar with two labels My Udhaari, My Account | 4 | 2 |
|  | Create My Udhaari component containing two radio buttons, purchase history and payment history and which will display a list of products purchased such as date, product, quantity, base price , total price of each product in tabular format | 10 | 8 |
|  | Create filters for month and year | 4 | 4 |
|  | Create an API to fetch and display the following details date, product, quantity, base price , total price from the database based on the filtered date | 10 | 5 |
|  | Routing using navigation to Payment History page with onPress() event handler | 4 | 1 |
| **TOTAL** |  | **32** | **20** |

Table 3.27 Payment Transaction

|  |  |
| --- | --- |
| **Sr. No.** | 27 |
| **User Story** | As a consumer, I want to, see only the payment transactions with total amount paid in a bill format on the screen |
| **Description** | Given I’m a successfully logged-in vendor (shopkeeper) and I’m on selected consumer’s ‘My Udhaari’ page  When I click on payment history of that consumer  Then the application shows a filter to see any years transaction, for easy access the application shows the paid amount, paid date along with the information of carry forwarded amount in bill format on clicking of it |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | onPress() of payment history create a payment history component, which displays a list of all previous payments paid | 4 | 4 |
|  | Create filters for year | 4 | 3 |
|  | Create an API to fetch  data such as the date on which payment was done and the total amount paid from the database | 8 | 6 |
|  | Routing using navigation to purchase history page | 4 | 1 |
| **TOTAL** |  | **20** | **14** |

Table 3.28 My Account

|  |  |
| --- | --- |
| **Sr. No.** | 28 |
| **User Story** | As a consumer, I want to, see all my account details for selected vendor (shopkeeper) on the screen |
| **Description** | Given I’m a successfully logged-in consumer  When I select ‘Account Details’ page  Then the application shows all my account details such as name, registered mobile number, account start date, threshold set by vendor (shopkeeper), last payment date, start date, due date, total due amount and partial due amount along with the status of my account whether it is active or blocked and vendors shop details |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create Account Details component for displaying account  details | 4 | 3 |
|  | Create an API to fetch  data such as name, RMN, account started date, threshold, last paid amount, start date, due date, total due amount and partial due amount | 10 | 13 |
|  | Create display label for showing active status | 4 | 3 |
|  | Routing using navigation to My Udhaari page and Logout | 4 | 1 |
| **TOTAL** |  | **22** | **20** |

Table 3.29 Back To Consumer Dashboard

|  |  |
| --- | --- |
| **Sr. No.** | 29 |
| **User Story** | As a consumer, I want to, go back to the Dashboard of the application |
| **Description** | Given I’m a successfully logged-in consumer  When I click on Home button  Then the application redirects me to the Dashboard  page |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a home button on different pages such as My Udhaari and Account Details page | 4 | 8 |
|  | Route using navigation with onPress() event handler | 4 | 2 |
| **TOTAL** |  | **8** | **10** |

Table 3.30 Consumer Logout

|  |  |
| --- | --- |
| **Sr. No.** | 30 |
| **User Story** | As a consumer I want to logout of the application |
| **Description** | Given I’m a successfully logged-in consumer  When I click on ‘Logout’ button  Then the application should redirect me to the Login page |

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Create a logout button on different pages such as account detail and my udhaari | 4 | 3 |
|  | Write an onPress() method to route using navigation to the Login page. | 4 | 3 |
| **TOTAL** |  | **8** | **6** |

Table 3.31 Development Estimation

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Total Of All Must Have User Stories** | **Time (units)** |
|  | Expected Time | 702 |
|  | Actual Time | 600 |

Requirements Estimation

This part involves, the estimated time required to complete the entire project including tasks other than the development (user stories).

* Requirements Gathering

1 unit = 30mins

Table 4.1 Requirement Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Categorized the roles - vendor (shopkeeper), consumer | 1 | 1 |
|  | Noted the requirements with actual personas | 6 | 7 |
|  | Noted the abstract features for the application | 8 | 7 |
|  | Comparison study with existing system and similar platforms | 8 | 8 |
|  | Finalising the features to be included in the application | 16 | 16 |
|  | Categorizing the features as must to have and nice to have features based on time constraints | 7 | 8 |
|  | Validate and analysing the requirements | 8 | 8 |
| **TOTAL** |  | **54** | **55** |

* Estimation (For Creating this document)

Table 4.2 Estimating Document Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task (Must Have + Nice To Have)** | **Expected Time (units)** | **Actual Time (units)** |
|  | Write down User stories | 8+2 | 16+4 |
|  | Description of each story | 24+4 | 24+4 |
|  | Test cases | 24+4 | 30+4 |
|  | Detailed task analysis | 48+8 | 48+8 |
|  | Peer Review | 48+8 | 44+12 |
| **TOTAL** |  | **152+26** | **162+32** |

* Design

Table 4.3 Design Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | Selecting a software development method to follow - Agile | 2 | 1 |
|  | Find entities, their attributes and relation between them | 4 | 2 |
|  | Create Entity Relationship diagram | 2 | 1 |
|  | Create database tables | 2 | 2 |
|  | Create diagrams - component, sequence, workflow, use case | 10 | 10 |
|  | Create must have features - wireframes | 16 | 16 |
|  | Select technology stack | 12 | 12 |
|  | Select target device | 10 | 10 |
|  | Create nice to have features - wireframes | 14 | 4 |
| **TOTAL** |  | **72** | **58** |

* Planning

Table 4.4 Planning Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.**  **No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | High level Planning | 16 | 16 |
|  | Coding Conventions | 10 | 4 |
|  | Day-wise task distribution among resources (Timeline Chart for tracking the tasks of each resource) | 24 | 16 |
|  | Review Planning | 4 | 4 |
| **TOTAL** |  | **54** | **40** |

* Study & Setup

Table 4.5 Study & Setup Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.**  **No.** | **Task** | **Expected Time (units)** | **Actual Time (units)** |
|  | React Native with Expo & Database Installations | 72 | 32 |
|  | Study React Native with Expo | 144 | 140 |
|  | Study Heroku (PostgreSql & hosting) | 72 | 70 |
|  | React External libraries, Create Repository | 48 | 32 |
| **TOTAL** |  | **336** | **274** |

* Testing

Table 4.6 Testing Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task (Must Have + Nice To Have)** | **Expected Time (units)** | **Actual Time (units)** |
|  | Functional Testing and User Interface (UI) Testing | 83+10 | 48 |
|  | Debugging | 48+12 | 45 |
| **TOTAL** |  | **131+22** | **93** |

* Documentation

Table 4.7 Documentation Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **Task No.** | **Task (Must Have + Nice To Have)** | **Expected Time (units)** | **Actual Time (units)** |
|  | Requirements Gathering | 18 | 20 |
|  | Estimation | 18 | 24 |
|  | Planning | 12 | 20 |
|  | Design | 12+6 | 30 |
|  | Development | 16+6 | 20 |
|  | Testing | 16+6 | 20 |
|  | Formatting the document | 8+4 | 25 |
| **TOTAL** |  | **116+22** | **159** |

Effort Estimation Report

This type of report describes, the efforts taken by the resources to estimate the time taken to compelete estimation for all the Software Development Life Cycle (SDLC) phases of the project.

* Comparison Report

Table 4.8 Comparison Report Of All SDLC Phase Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Tasks (Must Have + Nice To Have)** | **Expected Time (days)** | **Actual Time (days)** |
|  | Requirements Gathering | 4 | 4 |
|  | Estimation | 10+2 | 11+2 |
|  | Design | 5 | 4 |
|  | Planning | 4 | 3 |
|  | Study & Setup | 21 | 18 |
|  | Development | 41+12 | 38 |
|  | Testing | 9+2 | 6 |
|  | Documentation | 7+2 | 10 |
| **#** | **TOTAL ESTIMATED TIME** | **101+18** | **94+2** |

Total Time Estimated (Considering 8 hrs per working day)

Table 4.9 Total Estimation Report

|  |  |  |
| --- | --- | --- |
| **Features** | **Expected Time (days)** | **Actual Time (days)** |
| Only With Must To Have | 34 | 32 |
| Including Nice To Have | 40 | - |

Considering 3 resources and excluding the tasks completed

Table 4.10 Summary Of Estimation

|  |  |  |
| --- | --- | --- |
| **Features** | **Expected Date** | **Actual Date** |
| Only With Must Have | 10th May 2021 | 14th May 2021 |
| Including Nice To Have | 17th May 2021 | - |

Design

This phase describes, the architecture established. It starts with the requirement document delivered by the requirement phase and maps the requirements into an architecture.

Software Development Methodology

This section convey information about methodologies that play a vital part in developing a software. The basic purpose of these methodologies is to provide smooth software development according to the project requirements.

In this project, Agile software development methodology is used. According to this methodology ,the tasks are divided into time boxes (small time frames) to deliver specific features for a release. In this project each build is incremental in terms of features and  the final build holds all the features required by the client. Adaption of an agile methodology for the software development helps as it is suitable for fixed or changing requirements, easy to manage, gives flexibility to developers and hence forms a good model for environments that change steadily.

Peer review process

This is a type of review conducted so as to improve the productivity of the development team, to make the testing process time and cost effective, to make the final software with fewer defects and to eliminate the inadequacies. The following steps were conducted as a part of the peer review,

* Walkthrough - In this type each member of the development team are able to ask questions about code and make comments about defects.
* Code Review - This whole code is reviewed by all team members in a systematic way.

Collaboration Methodology

This methodology describes, the process of working together to come up with the product. It includes sharing resources with others which helps to get new ideas and also helps to find defects in the project.

Coding Guidelines

The guidelines set in this section describes, the coding standard to give a uniform appearance to the code written by different team members. It improves readability, and maintainability of the code and it reduces complexity also. It helps in code reuse and helps to detect error easily. For this project following are some coding guidelines set,

* Use prettier.io, font size - 14
* Use functional components ONLY
* Use arrow functions ONLY

E.g.,:

const SAMPLE = ()=> {

}

* File names - File (one word), File\_name(more than one word)
* PascalCase for function names
* Quotes - Use Single quotes everywhere
* Import statements in one line
* E.g.,:

import { StyleSheet, View, Image, TouchableOpacity, Alert } from 'react-native';

* camelCase for variables
* For constants - same pattern as file names
* Tab length - 4 spaces
* Add function description before each component including purpose, arguments, return value
* Convention for hooks - useHooks Functions
* Brackets – E.g.,:

FunctionName(x, y, z) {

Abc;

Cde;

}

* For stylesheets

E.g.,:

const styles = StyleSheet.create({

container: {

flex: 1,

justifyContent: ‘center’,

},

})

Source Code Repository

The current repository used for this project is GitHub. It is a collaborative coding tool with version control, branching and merging also having other features like, API, data import, file sharing, file transfer, data export, external integrations and multi-user. The link to this project’s repository is, “<https://github.com/anistamboli/Udhaari-Book>”

Technical stack details

This section provides more information about technology used in project such as, the solutions stack, technology infrastructure, list of all the technology services used to build and run this application.

Table 5.1 Technology Stack

|  |  |  |
| --- | --- | --- |
| **Technology** | **Versions** | **Reasons** |
| React Native   * minSDK - (21-25) * targetSDk | npm - 6.14.10  nodeJS - 14.15.4  Android Studio - 4.1.3 (lollipop 5.1, Portrait Mode)  JDK - 1.8.0\_202  Expo - 4.3.2 | * To explore more in Javascript, JSX. * Cross-platform to build mobile applications |
| Development Environment - Expo Client   * minSDK - 21 * latestSDK - 40 | PostgreSQL - 13.2 | * Easy platform for testing as QR code enables us to share apps from one device to another * It has integrated some basic libraries which can be useful for easy development |
| Hosting Services - Heroku |  | * Provides ready-to use environment that allows to deploy code fast |
| Online Databases - Heroku (PostgreSQL) |  | * Free cloud SQL database |
| Testing Device - Android 7+ versions  Portrait Mode |  | * As Expo client minSDK is 21 it supports only android 7+ versions * Online testing devices available support this version |

Technologies used, their versions

The following table lists the libraries with their versions used in this project,

Table 5.2 Technology, Libraries & Versions

|  |  |  |
| --- | --- | --- |
| **Package Name** | **Versions** | **Use** |
| navigation | 4.4.4 | Transition between screens and manage navigation history. |
| modal | 11.10.0 | An enhanced, customizable and animated pop model for react native |
| input-spinner | 1.7.10 | Enhance text input for entering  numeric values with increase and decrease button |
| gesture-handler | 1.8.0 | Provides native-driven gesture management APIs for building best possible touch-based experiences in React Native. |
| datetimepicker | 3.4.7 | Used to accept date or time in UI |
| react-navigation/material-top-tabs | 5.3.15 | Lets you switch between different routes by tapping the tabs or swiping horizontally |
| react-navigation/native | 5.9.4 | Navigate between screens |
| react-navigation/stack | 5.14.4 | Transition between screens where each new screen is placed on top of a stack. |
| axios | 0.21.1 | Make http calls using axios in react native |
| chai | 4.3.4 | Testing framework used for api testing |
| chai-http | 4.3.4 | Extend Chai Assertion library with tests for http apis |
| mocha | 8.4.0 | Testing framework used for api testing |
| express | 4.17.1 | A minimal and flexible Node.js web application framework that provides a robust set of features for mobile applications. |
| pg | 8.5.1 | Open source object-relational database |
| expo | 40.0.1 | Open-source platform for making universal native apps for Android |
| expo-secure-store | 9.3.0 | Provides a way to encrypt and securely store key–value pairs locally on the device |
| expo-status-bar | 1.0.3 | A component and imperative interface to control the app status bar to change its text colour, background colour, hide it, make it translucent or opaque, and apply animations to any of these changes |
| react | 16.13.1 | React  render on the server using Node and power mobile apps using [React Native](https://reactnative.dev/) |
| react-dom | 16.13.1 | Provides DOM-specific methods that can be used at the top level of  app |
| elements | 3.3.2 | Provides an all-in-one UI kit for creating apps in react native |
| safe-area-context | 3.1.9 | Provides a flexible API for accessing device safe area inset information |
| reanimated | 1.13.0 | Provides an API that greatly simplifies the process of creating smooth, powerful, and maintainable animations |
| react-native-screens | 2.15.2 | Provides native primitives to represent screens |
| react-native-tab-view | 2.16.0 | A cross-platform Tab View component for React Native |
| react-native-web | 0.13.12 | React Native Web makes it easy to bring your React Native app to the web |

Database Design

This part describes, a collection of steps that help with designing, creating, implementing, and maintaining data management system required for the application. The main purpose of designing a database is to produce physical and logical models of designs for the proposed system.

ER Diagrams

The diagram which provides simple representation of the structure of a database with the help of a diagram is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that is implemented as a database. The main components of E-R model are: entity set and relationship set listed as follows,

* This diagram shows the relationship between entities such as vendor (shopkeeper), consumer, product, payment\_history and transaction\_history in the system.
* The system enables multiple vendors to add many consumers so the relationship between vendor (shopkeeper) and consumer is many-to-many.
* Along with this, one vendor (shopkeeper) can handle payment history of multiple consumers and one consumer can view all their payment history hence their relationship is one-to-many.
* Multiple vendors can have multiple products and multiple consumer can select multiple products hence their relationship is many-to-many .
* All transactions of consumers are stored in transaction\_history, where one vendor (shopkeeper)\_consumer can have multiple transactions .Hence the relationship between transaction\_history and vendor (shopkeeper)\_consumer  is one to many.
* Transaction history  has 2 types namely, payment type and purchase type that means one transaction history can have many payment and purchase history. Hence the relationship between transaction\_history and payment\_history and consumer\_product\_vendor (purchase\_history) is one to many.

Diagram

Description automatically generated

Figure 2 ER Diagram

Entity Description

This part of the project describes, the list of entities involved in this project are as follows,

* vendor (shopkeeper)

This database table stores the details of all registered vendors. Each row is  identifiable via contact number which is used as the primary key for each user. Vendor (shopkeeper)’s name, shop name, shop address and password are also required fields.

Table 6.1 Database Tables - vendor

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| contact | BIGINT | PRIMARY KEY |
| name | VARCHAR (30) | NOT NULL |
| shop\_name | VARCHAR (30) | NOT NULL |
| shop\_address | VARCHAR (50) | NOT NULL |
| password | VARCHAR (20) | NOT NULL |

* consumer

This database table stores the details of all registered consumers. Each row is identifiable via contact number which is used as the primary key for the user. Consumer’s name, address and password are also required fields.

Table 6.2 consumer

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| contact | BIGINT | PRIMARY KEY |
| name | VARCHAR (30) | NOT NULL |
| address | VARCHAR (50) | NOT NULL |
| password | VARCHAR (20) | NOT NULL |

* vendor (shopkeeper)\_consumer

This database table stores the details of all Udhaari accounts created by vendors and consumers by many to many relation i.e. any vendor (shopkeeper) can have any consumer’s Udhaari and vice-versa. Since it has many to many relations it consists of two foreign keys: vendor's contact number from vendor table and consumer’s contact number from consumer table. Threshold and balance are set to 0 by default and the vendor can store any value between 0 and 1 for threshold and any other value for balance in it. Start date and due date of the consumer's account are also stored.

Table 6.3 vendor\_consumer

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| vendor (shopkeeper)\_contact | BIGINT | FOREIGN KEY |
| consumer\_contact | BIGINT | FOREIGN KEY |
| balance | FLOAT | DEFAULT 0 |
| threshold | FLOAT | CHECK (0<threshold<1), DEFAULT 0 |
| start\_date | DATE | NOT NULL |
| due\_date | DATE | NOT NULL |
| billing\_start\_date | DATE | NOT NULL |
| consumer\_name | VARCHAR (30) | NOT NULL |

* product

This database table stores the details of all products. Each row is  identifiable via id which is used as the primary key for each product. Name and base price of products are also required fields.

Table 6.4 product

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| Id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| Name | VARCHAR (50) | NOT NULL |
| base\_price | FLOAT | NOT NULL |

* consumer\_product\_vendor

This database table stores the records of all purchase history. Each row is  identifiable via id which is used as the primary key for each purchase. There is a many to many relation between three entities i.e. any consumer buys product/s from any vendor (shopkeeper). Since it has many to many relations it consists of three foreign keys: vendor's contact number from vendor table, consumer’s contact number from consumer table and product id from product table. It also stores the quantity and date-time of the purchase.

Table 6.5 consumer\_product\_vendor

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| Id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| vendor (shopkeeper)\_contact | BIGINT | FOREIGN KEY |
| consumer\_contact | BIGINT | FOREIGN KEY |
| product\_id | INT | FOREIGN KEY |
| tr\_id | BIGINT | FOREIGN KEY |
| quantity | FLOAT | NOT NULL |
| date\_purchase | DATE | now(), NOT NULL |
| time\_purchase | DATETIME | now(), NOT NULL |
| total\_price | FLOAT | NOT NULL |

* payment\_history

This database table stores the details of all transactions done by the consumers through vendors by many to many relation i.e. any vendor (shopkeeper) can add payment of any consumer’s Udhaari to update balance. Each row is identifiable via id which is used as the primary key for each payment. Since it has many to many relations it consists of two foreign keys: vendor's contact number from vendor table and consumer’s contact number from consumer table. It also stores the payment details like total amount, paid amount, remaining amount and date-time of the transaction.

Table 6.6 payment\_history

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| consumer\_contact | BIGINT | FOREIGN KEY |
| vendor (shopkeeper)\_contact | BIGINT | FOREIGN KEY |
| tr\_id | BIGINT | FOREIGN KEY |
| total\_amount | FLOAT | NOT NULL |
| payed\_amount | FLOAT | NOT NULL |
| remaining\_amount | FLOAT | NOT NULL |
| transaction\_date | DATE | now() |
| transaction\_time | DATETIME | now() |

* transaction\_history

This database table stores the details of all transactions such as payment history or purchase history done by the consumers through vendors by many to many relation i.e. any vendor (shopkeeper) can add payment of any consumer’s Udhaari to update balance or vendor can see the list of purchase products. Each row is identifiable via id which is used as the primary key for each payment. Since it has many to many relations it consists of two foreign keys: vendor's contact number from vendor table and consumer’s contact number from consumer table. It also stores the payment details like  type , transaction amount and date-time of the transaction.

Table 6.7 transaction\_history

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Constraints** |
| id | INT | PRIMARY KEY, AUTO\_INCREMENT |
| consumer\_contact | BIGINT | FOREIGN KEY |
| vendor (shopkeeper)\_contact | BIGINT | FOREIGN KEY |
| type | VARCHAR(20) | NOT NULL |
| transaction\_amount | FLOAT | NOT NULL |
| transaction\_date | DATE | now() |
| transaction\_time | DATETIME | now() |

Work Flow Diagram

This diagram shows the graphical view of the system. It highlights step by step and linear representation of the project from start to finish and also depicts the exact steps required to complete a project

A picture containing timeline

Description automatically generated

Figure 3 Workflow Diagram

Component Diagram

This diagram consists of all the components required for the system. The user interface components in the system are as follows,

* Vendor (shopkeeper)
* Consumer
* Product
* Udhaari Records
* Payment
* Details

It also contains a Database component. This diagram shows the relation between all the components derived.

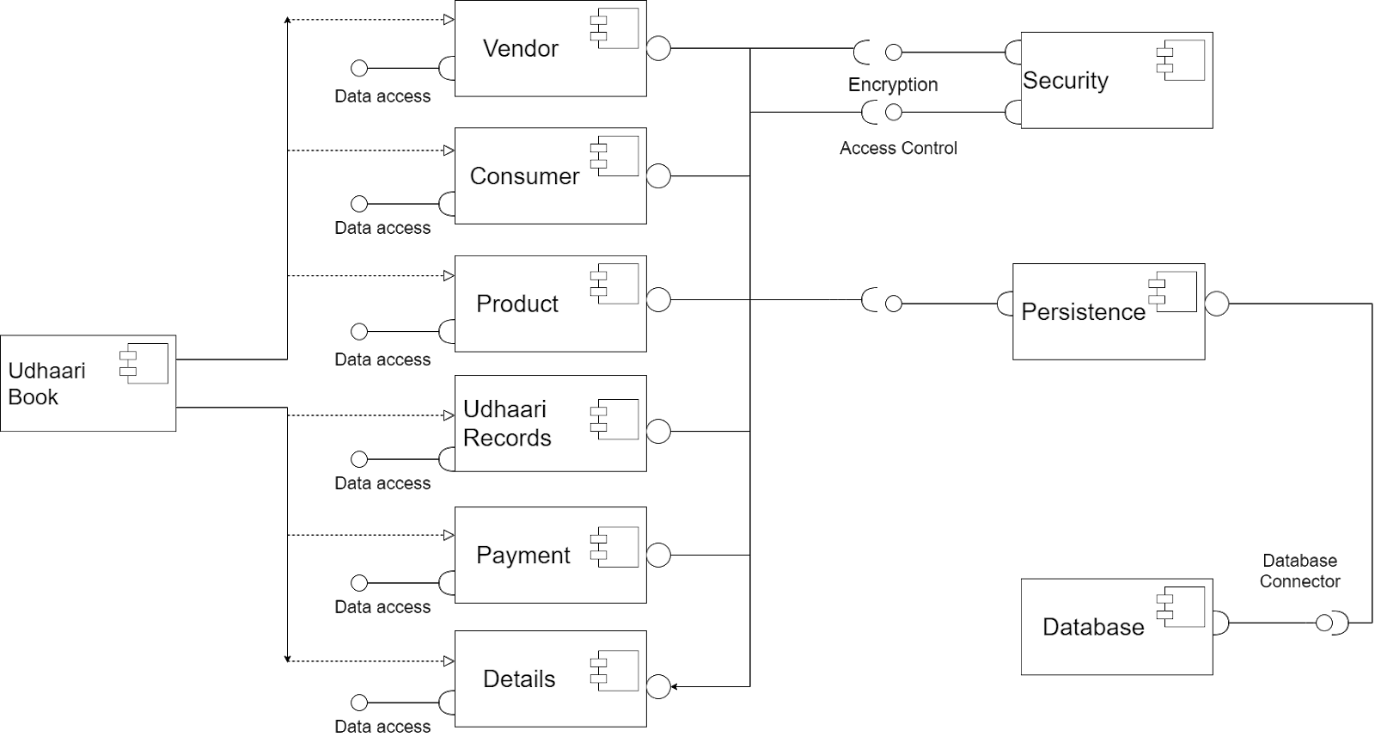


Figure 4 Component Diagram

Sequence diagrams

* Vendor (shopkeeper)

These diagrams are used to represent the flow and messages among each component  and also depicts the dynamic scenarios of the project. This diagram describes how vendors and other components are related with each other and in what order they communicate with each other.

Shape, rectangle

Description automatically generated with medium confidence

Figure 5.1 Vendor Sequence Diagram

* Consumer

This sequence diagram describes how consumers and other components are related with each other and in what order they communicate with each other.

Graphical user interface

Description automatically generated

Figure 5.2 Consumer Sequence Diagram

Wireframes

This section describes, layout of a web page that demonstrates what interface elements will exist on key pages. This system consists of the following wireframes,

* Logo/Launching Screen

This screen contains the name of the application along with the two categories vendor (shopkeeper) and consumer through which users can register / login based on their roles.

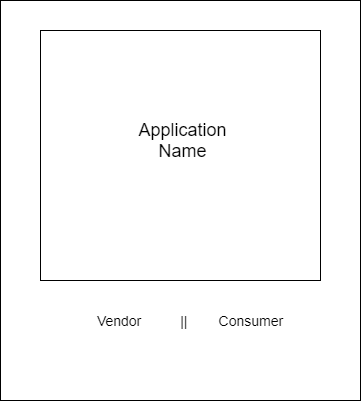
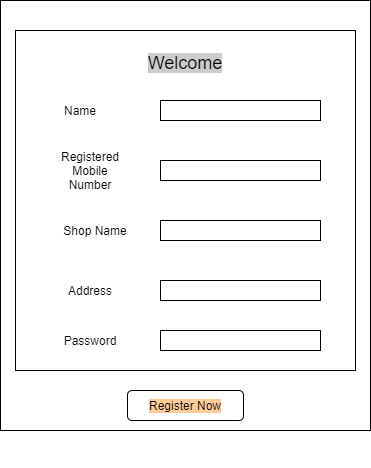


Figure 6.1 Wireframes - Launching Screen

Vendor (shopkeeper) Register

This screen contains a registration form for an unregistered user who is a vendor

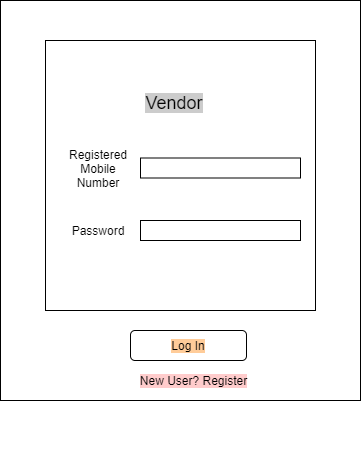
Figure 6.2 Vendor Registration



* Vendor (shopkeeper) Login

This screen contains a login form for a registered vendor (shopkeeper) through which the Dashboard can be accessed. It also provides a link to the registration form for new vendors

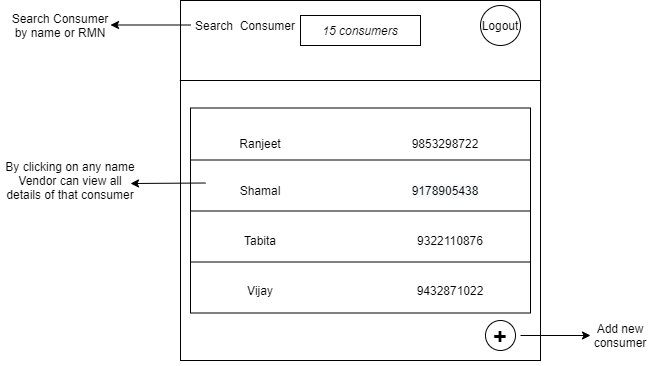
Figure 6.3 Vendor Login



* Vendor (shopkeeper) Dashboard

This screen contains the list of all consumers taken udhaari from the logged in vendor (shopkeeper) along with a search bar to search for any particular consumer, floating button to add a new consumer to consumers list and a logout button

Figure 6.4 Vendor Dashboard



* Add New Consumer (onclick of floating button)

This screen contains a search bar through which vendors can search for all registered consumers in order to add them to their udhaari,. The fields of this form can be filled only by the logged in vendor (shopkeeper) for each new consumer and add them to their consumers list

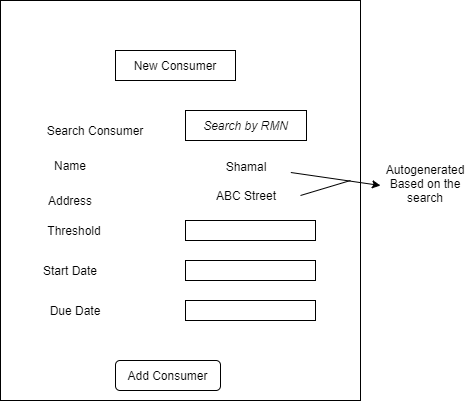
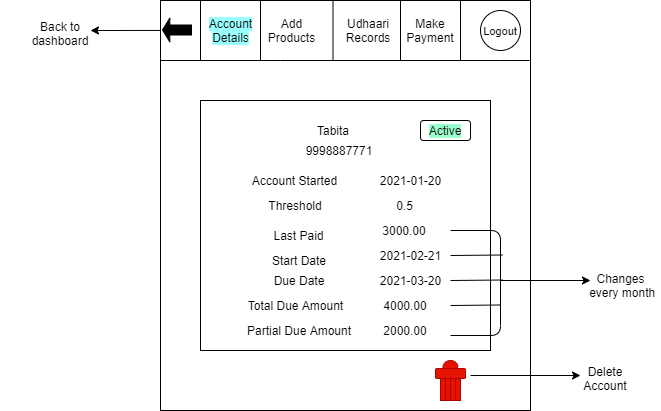


Figure 6.5 Add New Consumer

* Account Details (Onclick of any consumer)

This is the first screen that is rendered after the vendor (shopkeeper) selects a consumer from their consumers list and is also the first option on the navbar. This screen contains the account details of that consumer and also shows the current status of the account along with a delete account option. The navbar contains a back button to go back to the Dashboard and the logout button.

Figure 6.6 Account Details



* Add Products

This is the second option on the navbar. This screen contains a table through which vendors can add details of purchased products for the selected consumer to keep a record of it. It auto calculates and displays the total price for each product as well as the total amount.

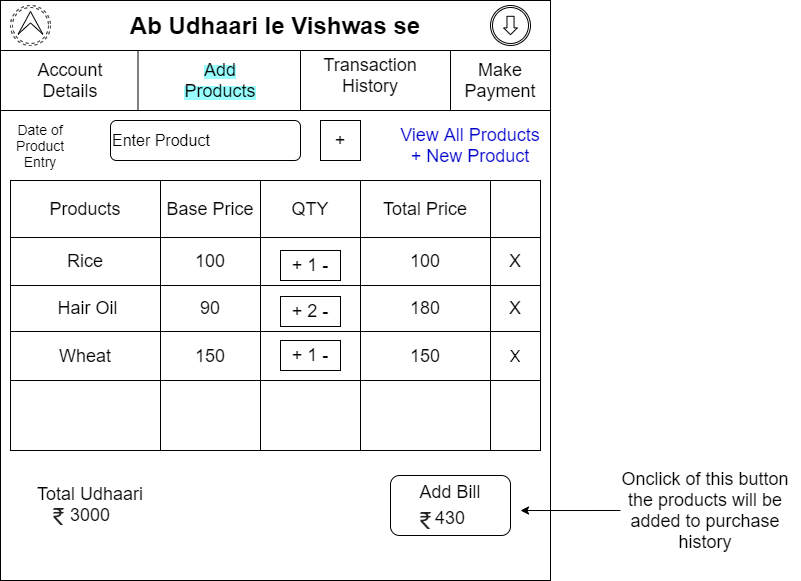


Figure 6.7 Add Products

* View All Products

This is the option on the right corner in the Add Products page. In this vendor (shopkeeper) can see the list of all products with their name and base price . Vendors can also add new products to the list.



Figure 6.8 View Products

* Transaction History

This is the third option on the navbar. This screen contains two options: purchase  and payment. By default this page shows all transactions of purchase as well as payment. By clicking on any purchase or payment card vendor (shopkeeper) can view detail information about it.

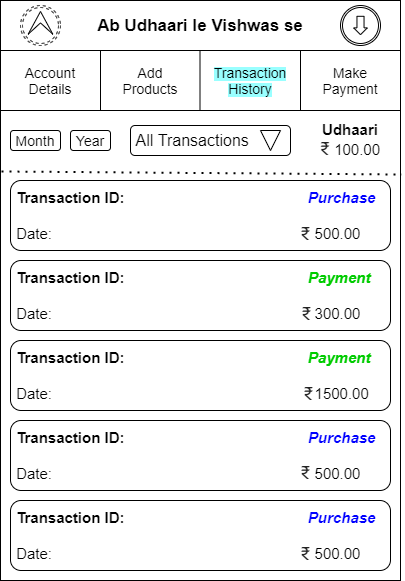


Figure 6.9 Transaction History

* Payment

This is the option available in Transaction record.By clicking on payment vendor (shopkeeper) can view all details of payment such as transaction\_Id , date , total amount , paid amount and carry forwarded amount of particular consumer.

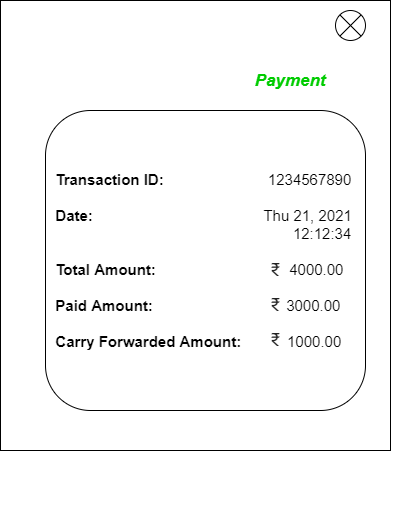


Figure 6.10 Payment Transaction

* Purchase

This is the option available in Transaction record.By clicking on purchase list, vendor (shopkeeper) can view list of all purchase products with their transaction\_Id, Date, Product, base price, quantity and total.

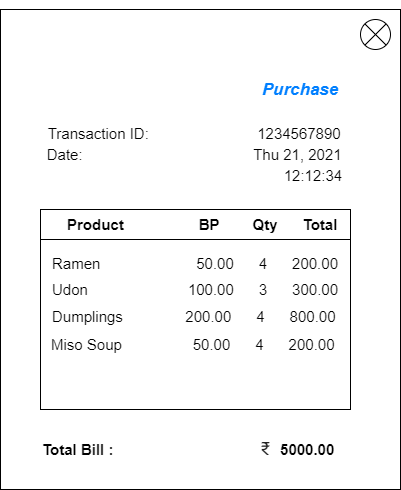


Figure 6.11 Purchase Transaction

* Make Payment

This is the fourth navbar option. This screen contains payment options through which vendors can manually clear the selected consumer’s bill and update their balance.

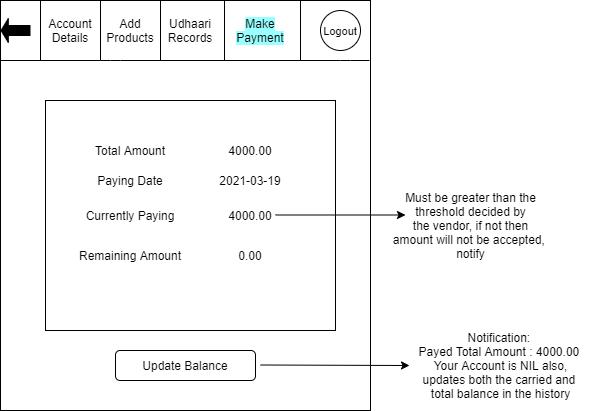
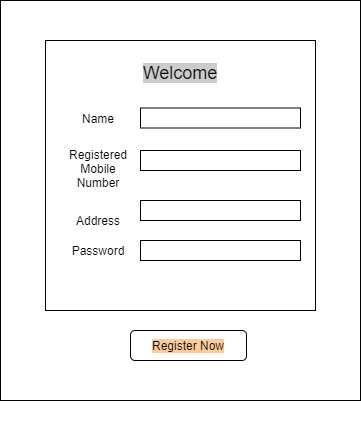


Figure 6.12 Make Payment

* Consumer Registration

This screen contains a registration form for an unregistered user who is a consumer. Only a registered consumer can take udhaari/credit from any vendor (shopkeeper).

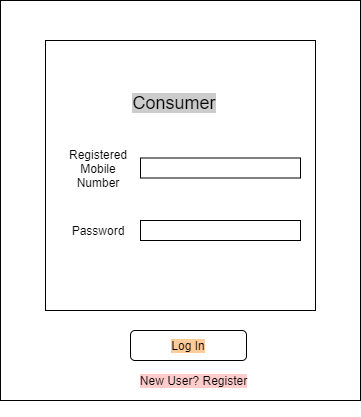
Figure 6.13 Consumer Registration



* Consumer Login

This screen contains a login form for a registered consumer through which the Dashboard can be accessed. It also provides a link to the registration form for new consumers.

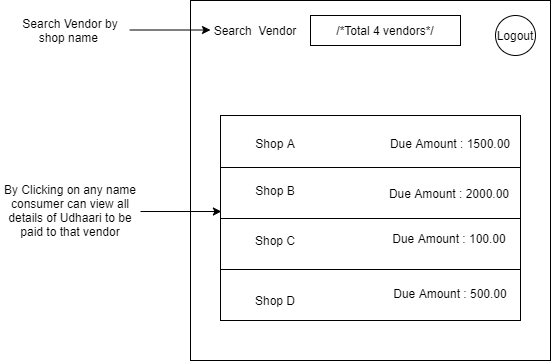
Figure 6.14 Consumer Login



* Consumer Dashboard

This screen contains the list of all vendors shop with the due amount to be paid to that vendor (shopkeeper) from which the logged in consumer has taken udhaari along with a search bar to search for any particular shop and a logout button

Figure 6.15 Consumer Dashboard



* Transaction History

This is the third option on the navbar. This screen contains two options: purchase  and payment. By default this page shows all transactions of purchase as well as payment. By clicking on any purchase or payment card vendor (shopkeeper) can view detail information about it.

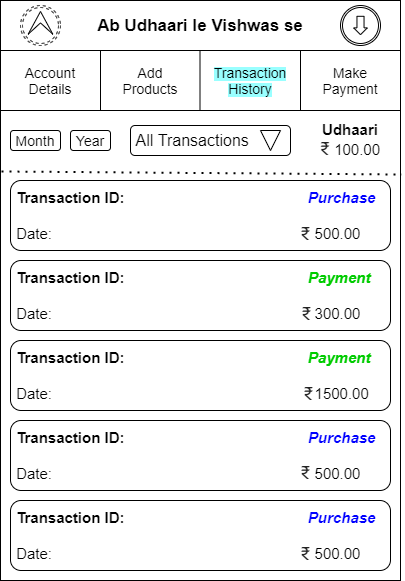


Figure 6.16 Transaction History

* Payment

This is the option available in Transaction record.By clicking on payment vendor (shopkeeper) can view all details of payment such as transaction\_Id , date , total amount , paid amount and carry forwarded amount of particular consumer.

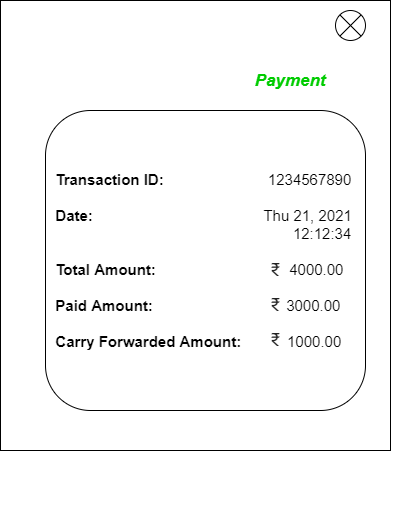


Figure 6.17 Payment Transaction

* Purchase

This is the option available in Transaction record.By clicking on purchase list, vendor (shopkeeper) can view list of all purchase products with their transaction\_Id, Date, Product, base price, quantity and total.

###### 

Figure 6.18 Purchase Transaction

* My Account

This is the second option on the navbar. This screen contains the account details of the logged in consumer and also shows the current status of their account related to the selected vendor (shopkeeper)

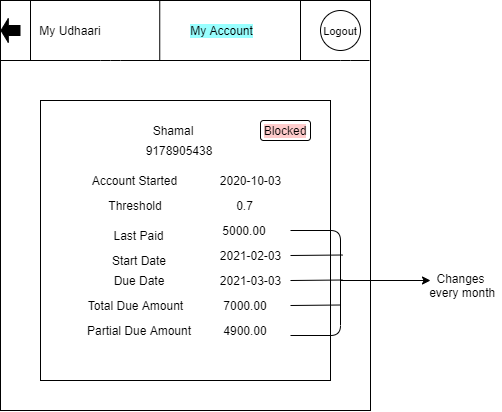


Figure 6.19 My Account

Development

This section gives information about, the [structure](https://en.wikipedia.org/wiki/Structure) and control process of the actual development of the system. It involves coding and debugging the application functionalities.

Application flow

This is the actual flow of the application, wherein the two roles vendor (shopkeeper) and consumer have different functionalities because of which the flow of the application for each differs and are as follows,

* Vendor (shopkeeper)

Vendor needs to login to the application. If vendor is not registered then vendor has to register first to access the system. Once vendor is logged-in, they are redirected to the vendor’s dashboard, which is also the homepage for the vendor. Now vendor can perform following actions,

* Search for a particular consumer by their name or registered mobile number
* By clicking on floating button vendor can add new consumer
* Vendor can view a list of all consumers on the dashboard
* Vender can view all details of selected consumer on account details page
* Name and threshold of consumer can be edited by vendor
* Vendor can delete the account of selected consumer
* Vendor can add list of products purchased as udhaari by particular consumer
* Vendor can view total bill of purchased products as well as total due udhaari of that consumer
* Vendor can view the list of all products with their name and base price
* Vendor can also add a new product to the list of products
* All transactions of particular consumer can be viewed by vendor in udhaari records
* Vendor can view purchase bill of selected consumer
* Vendor can view payment bill of selected consumer
* Payment details can be updated by vendor for consumer using make payment page
* Vendor can go back to homepage by clicking on home button
* Vendor can logout of the application by clicking on logout button
* Consumer

Consumer needs to login to the application. If consumer is not registered then consumer has to register first to access the system. Once consumer is logged-in, they are redirected to the consumer’s dashboard, which is also the homepage for the consumer. Now consumer can perform following actions,

* Search for a particular vendor (shopkeeper) by their shop name or registered mobile number
* Consumer can view a list of all vendors on consumers dashboard
* Consumer can view all details of their account related to the selected vendor on Account Details page
* Consumer can view total bill of purchased products as well as total due udhaari to that vendor
* All transactions the consumer is avaialable in my udhaari page
* Consumer can view their purchase bill
* Consumer can view their  payment bill
* Consumer can go back to homepage by clicking on home button
* Consumer can logout of the application by clicking on logout button

Application features

It involves, the actual features and functionalities of this application along with the actual screen shots of each page developed

Launching Screen

This is the first screen users land on when they open this application. It contains the application name and the categorized two roles for users . User has to select one of the a roles to proceed and enter login credentials to access the application. If user is not registered then it is mandatory to register first in order to login and access the further features.

Logo, company name

Description automatically generated

***Figure 7.1 Screen Shots – Launching Page***

Vendor (shopkeeper) Login

This is the screen designed for vendor to login, if vendor is already registered then he/she can login directly with their login credentials. System checks login credentials if contact number and password is valid then system redirects the vendor to dashboard otherwise system provides validation error message.

Graphical user interface, website

Description automatically generated

***Figure 7.2 Vendor Login***

Vendor (shopkeeper) Register

* This is the screen designed for vendor to register. If a vendor is unregistered and wants to login then he/she has to register first by their name, contact number, shop name, shop address and password. System checks validations if all fields are correct then system registers the new vendor and allows vendor to login into the system.

Graphical user interface, text, application

Description automatically generated

***Figure 7.3 Vendor Registration***

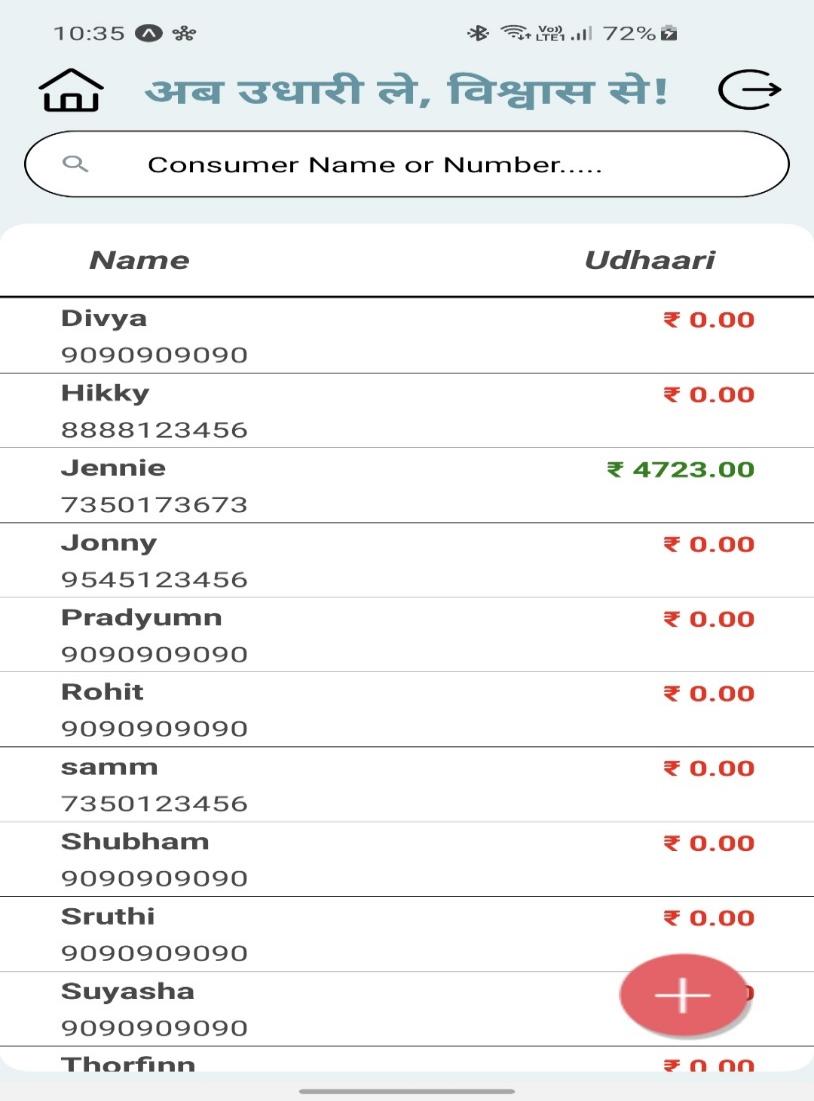
* If Vendor enters all valid credentials then system gives message of registration successful and allows the vendor to login through their registered mobile number and password.



***Figure 7.4 Vendor Registration Successful***

* Vendor (shopkeeper) Dashboard

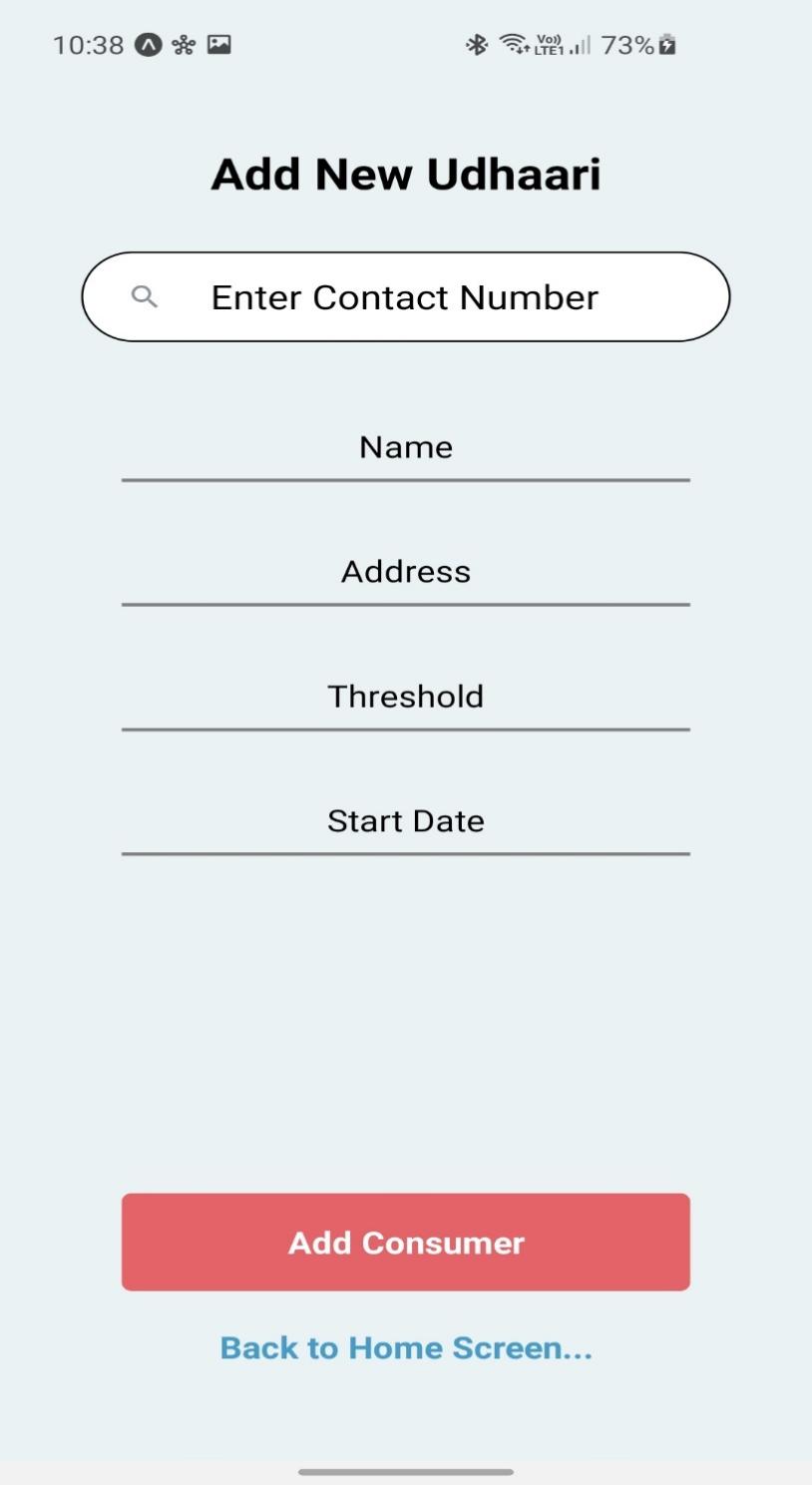
This is the screen designed as dashboard for a vendor. If vendor is successfully logged-in he/she will be redirected to this page. This itself is the homepage of the vendor. This page contains a search bar to search for a particular consumer by their name or registered mobile number along with this there is the list of all consumers taken udhaari/credit with their name, registered mobile number and total due amount. There is a floating button through which vendors can add new consumers to the list. Vendor can also logout from system by clicking on logout button.



***Figure 7.5 Vendor Dashboard***

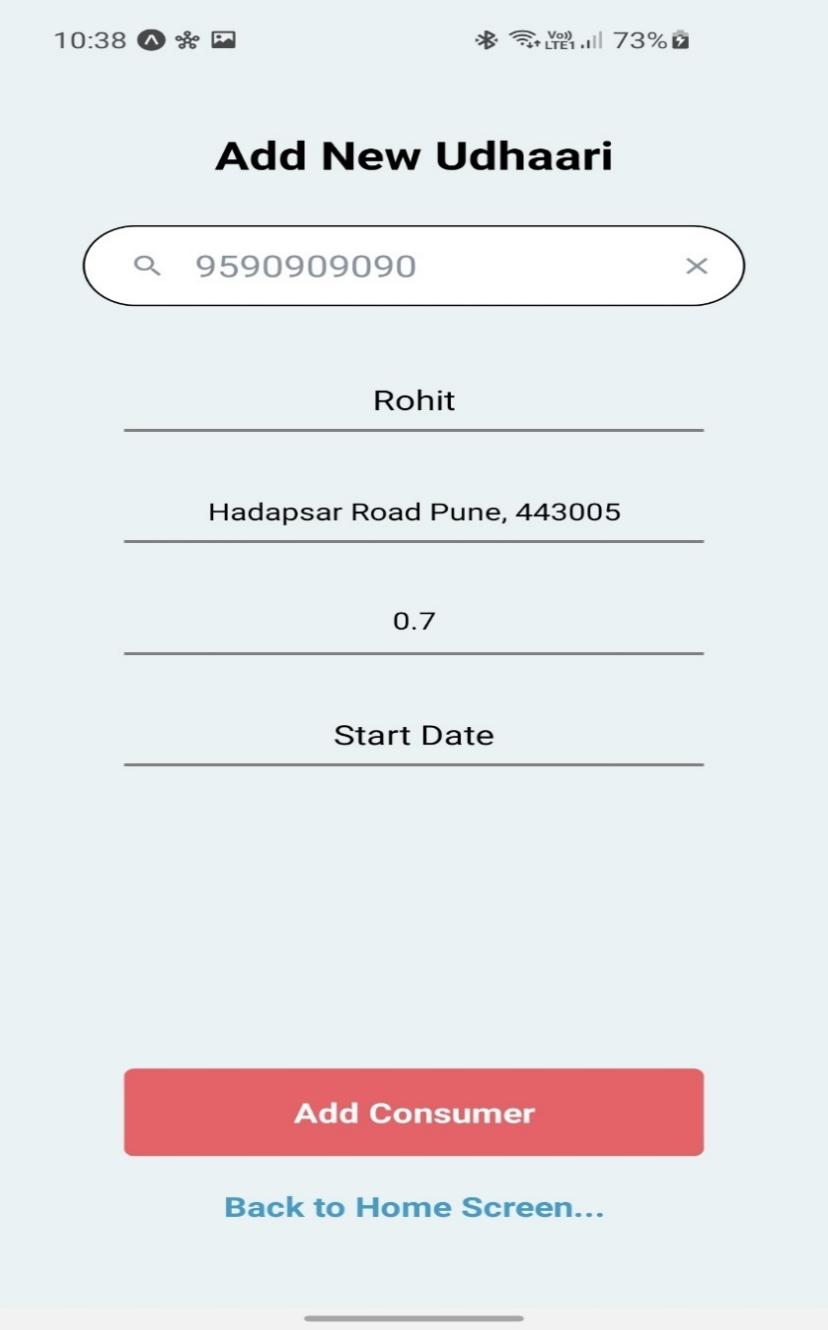
Add New Consumer

* This is a the add new consumer screen where vendor can add new consumer by searching their registered mobile number in the search bar.



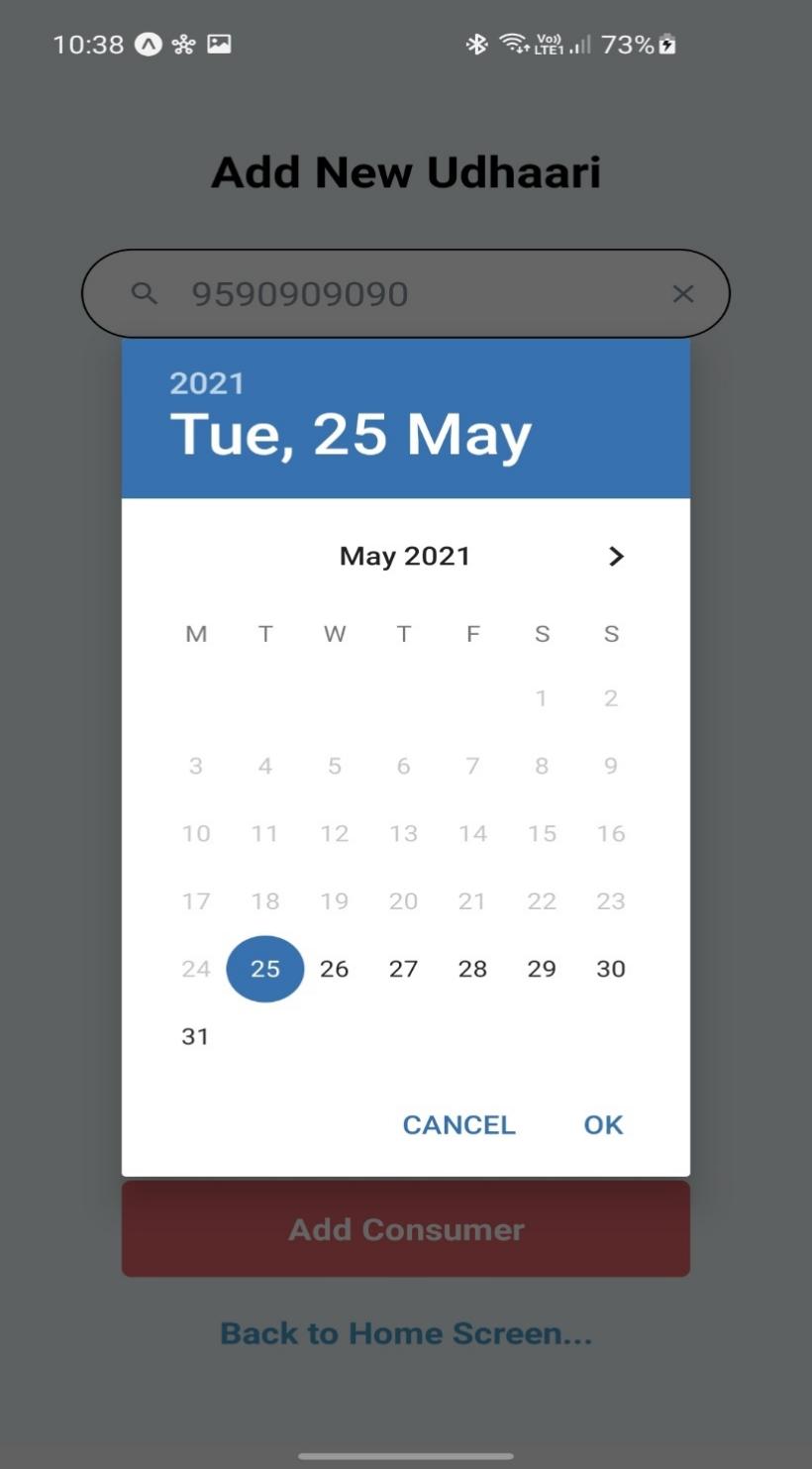
***Figure 7.6 Add New Consumer***

* Vendor (shopkeeper) searches for particular consumers with their registered mobile number. If a consumer is registered in the system, then system displays name and address of searched consumer. If the contact number is not registered then system displays empty array of name and address.



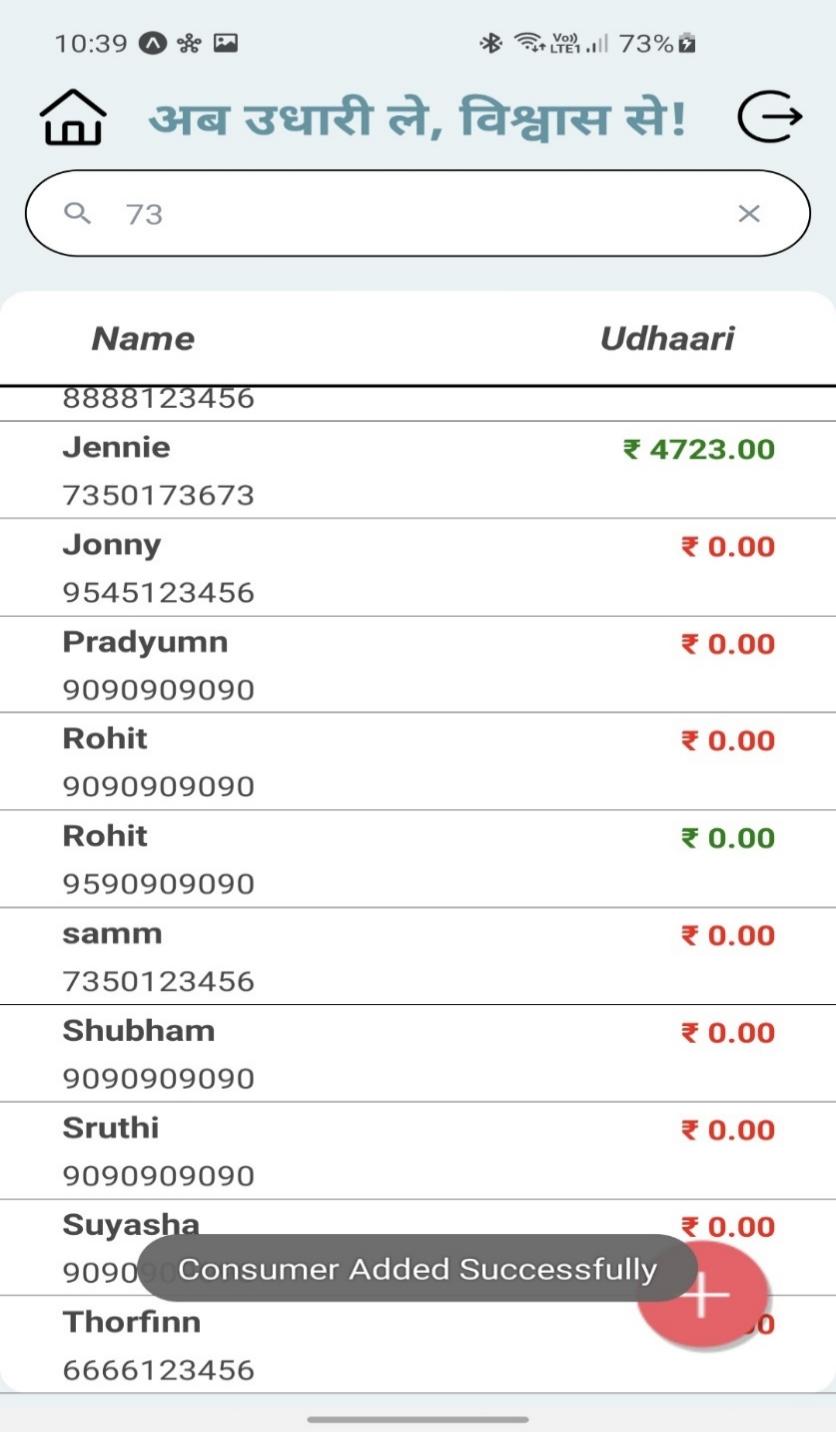
***Figure 7.7 Search To Add A Registered Consumer***

* Vendor (shopkeeper) can set the start date for the account of that consumer. Based on start date, due date is automatically set with a span 30 days from start date.



***Figure 7.8 Set Account Start Date***

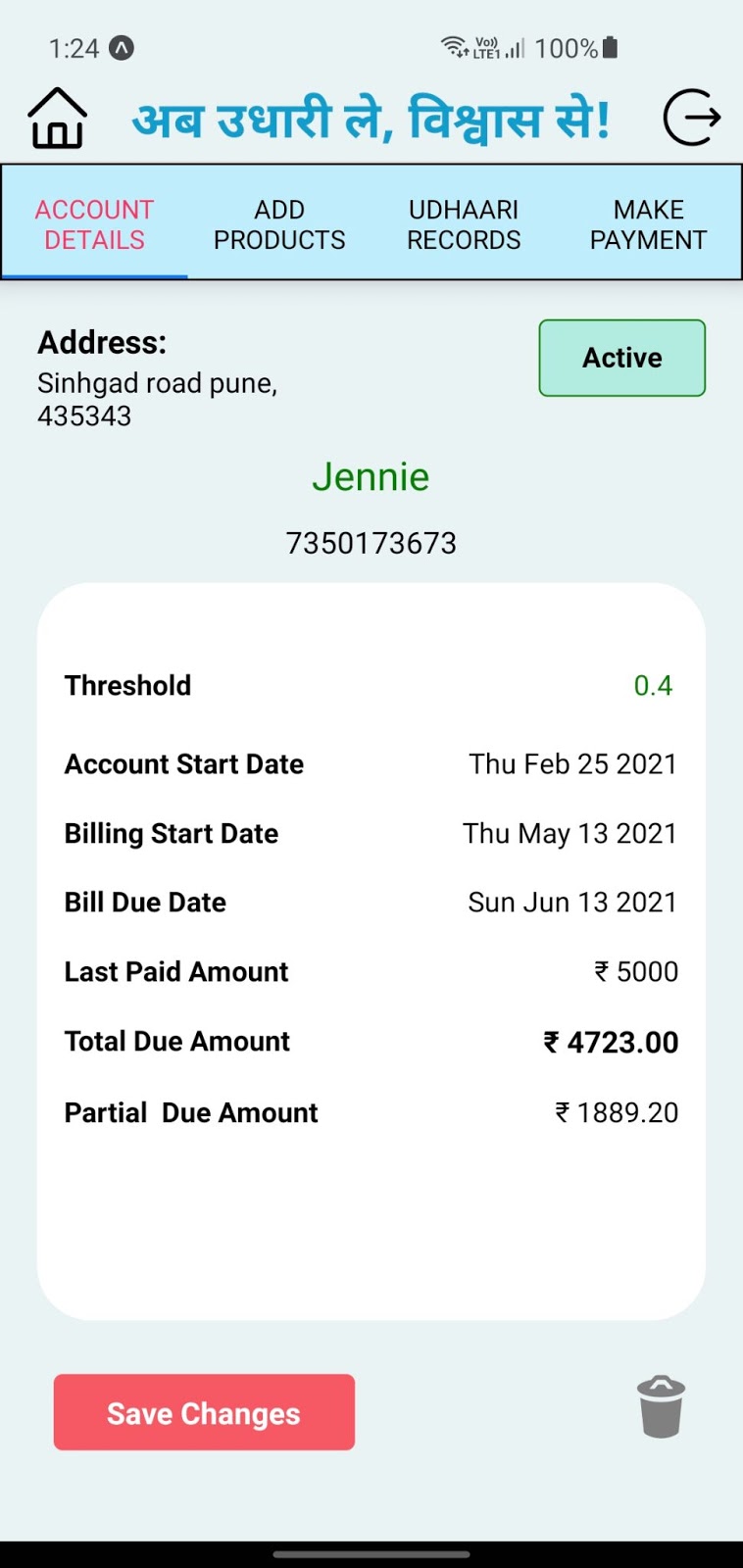
* If the consumer is already registered and if vendor (shopkeeper) enters all valid data for threshold and start date then system allows vendor to add new consumer and gives message “Consumer added successfully” and redirects the vendor to dashboard.



***Figure 7.9 Consumer Added Successfully***

Vendor (shopkeeper) Account Details

* When the vendor selects any consumer from the dashboard , the vendor is redirected to the Account details page where the vendor can view all details of a selected consumer.



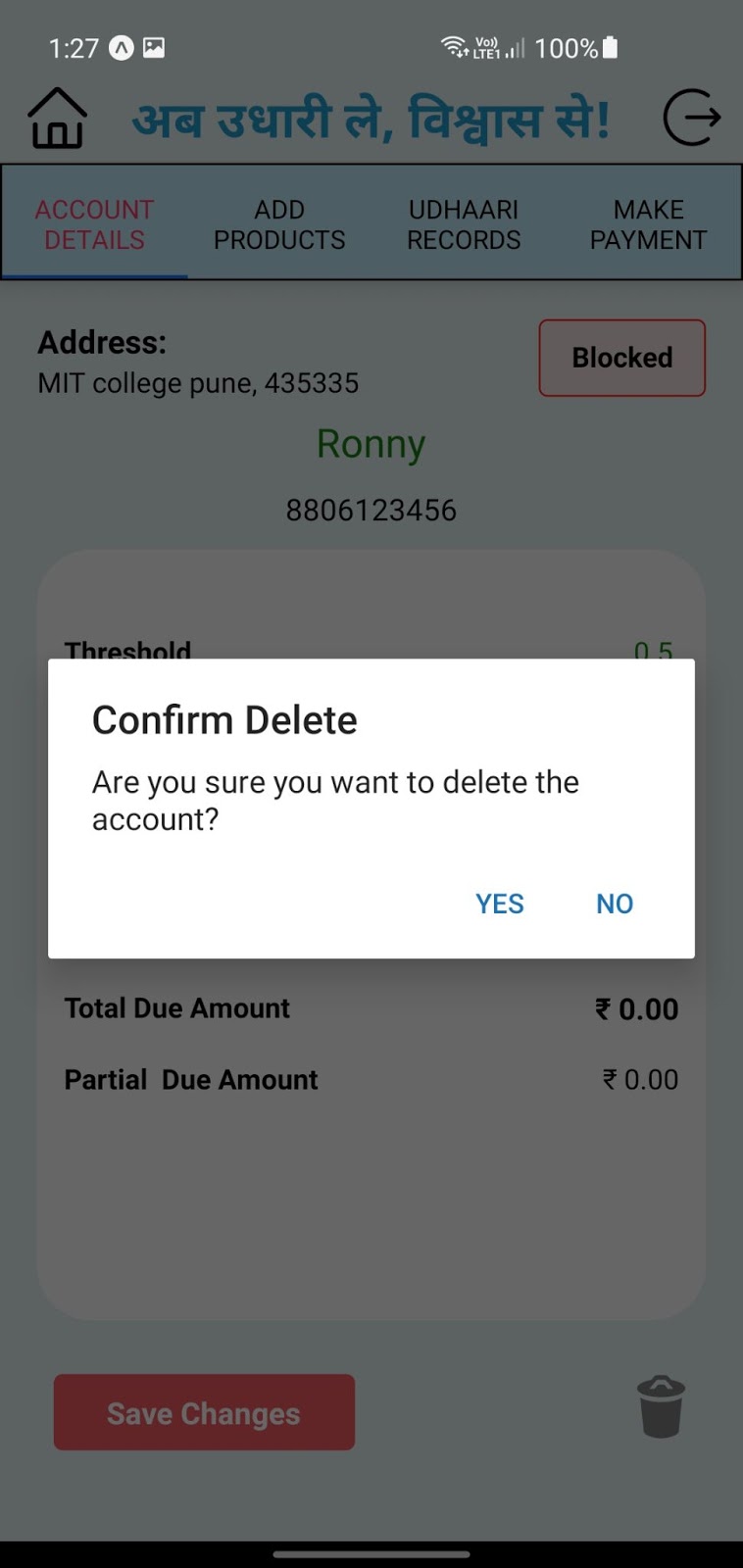
***Figure 7.10 Account Details Of Selected Consumer***

* Vendor (shopkeeper) can also edit name and threshold for particular vendor and click on save changes button then system saves the new changes and display message “Name changed successfully” or “Threshold change successfully” accordingly.



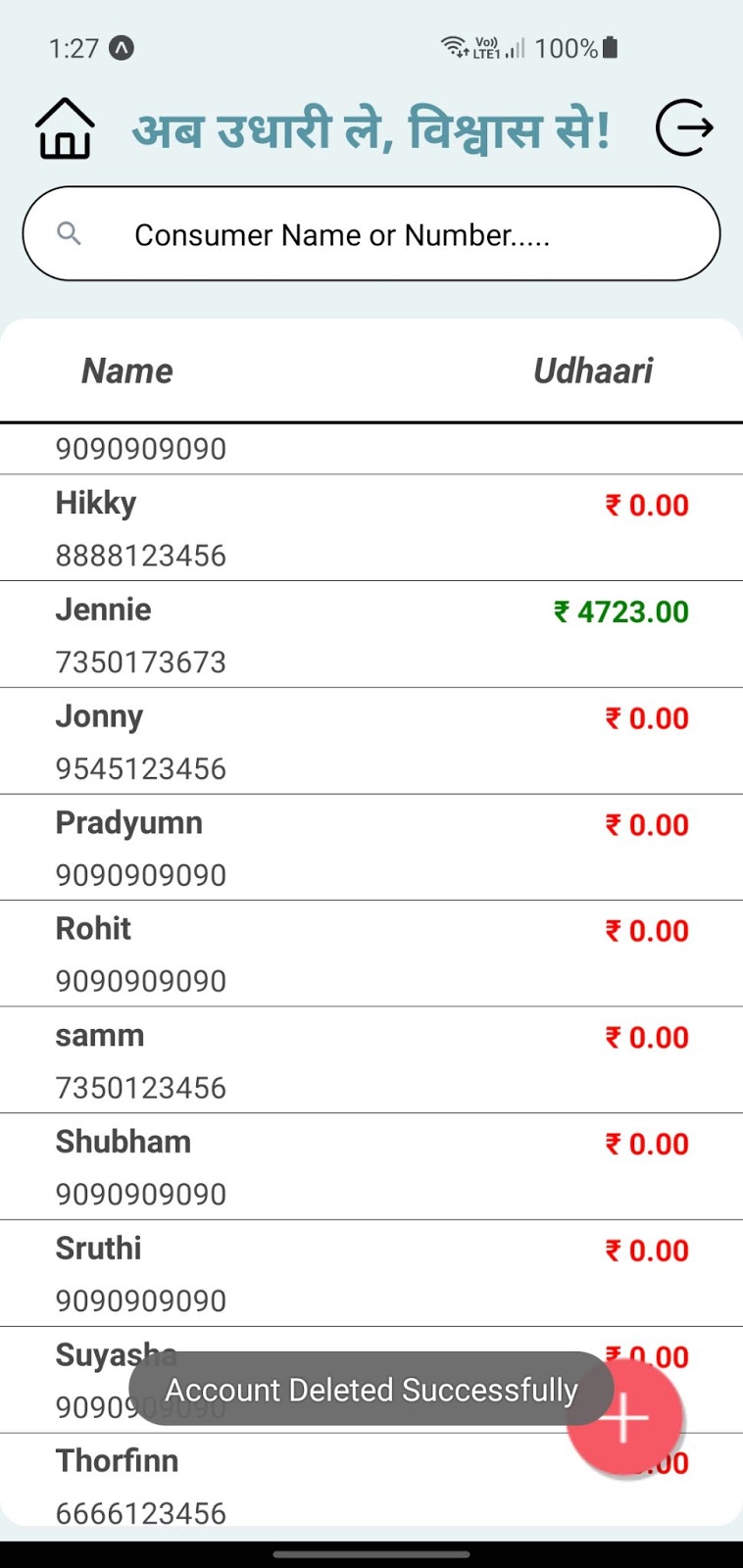
***Figure 7.11 Edit Account***

* If vendor (shopkeeper) wants to delete a particular account, then vendor clicks on delete icon, system first confirms from vendor to delete the account if vendor clicks yes then system deletes the particular account from that vendor’s list.



***Figure 7.12 Delete Account***

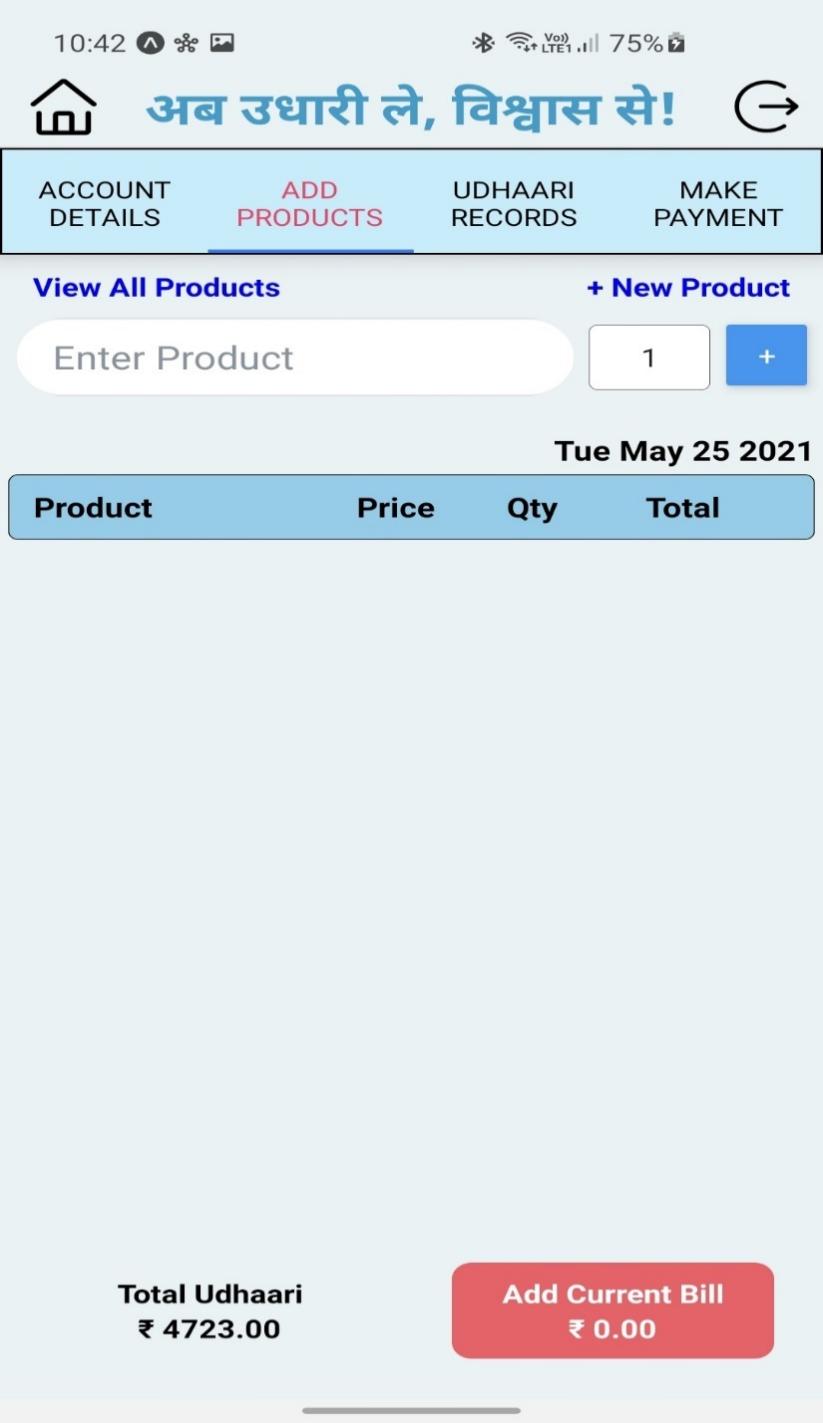
* If vendor (shopkeeper) clicks yes to delete account  then system deletes particular account from that vendor’s list and redirects the vendor to dashboard along with message “Account deleted successfully”.



***Figure 7.13 Account Deleted Successfully***

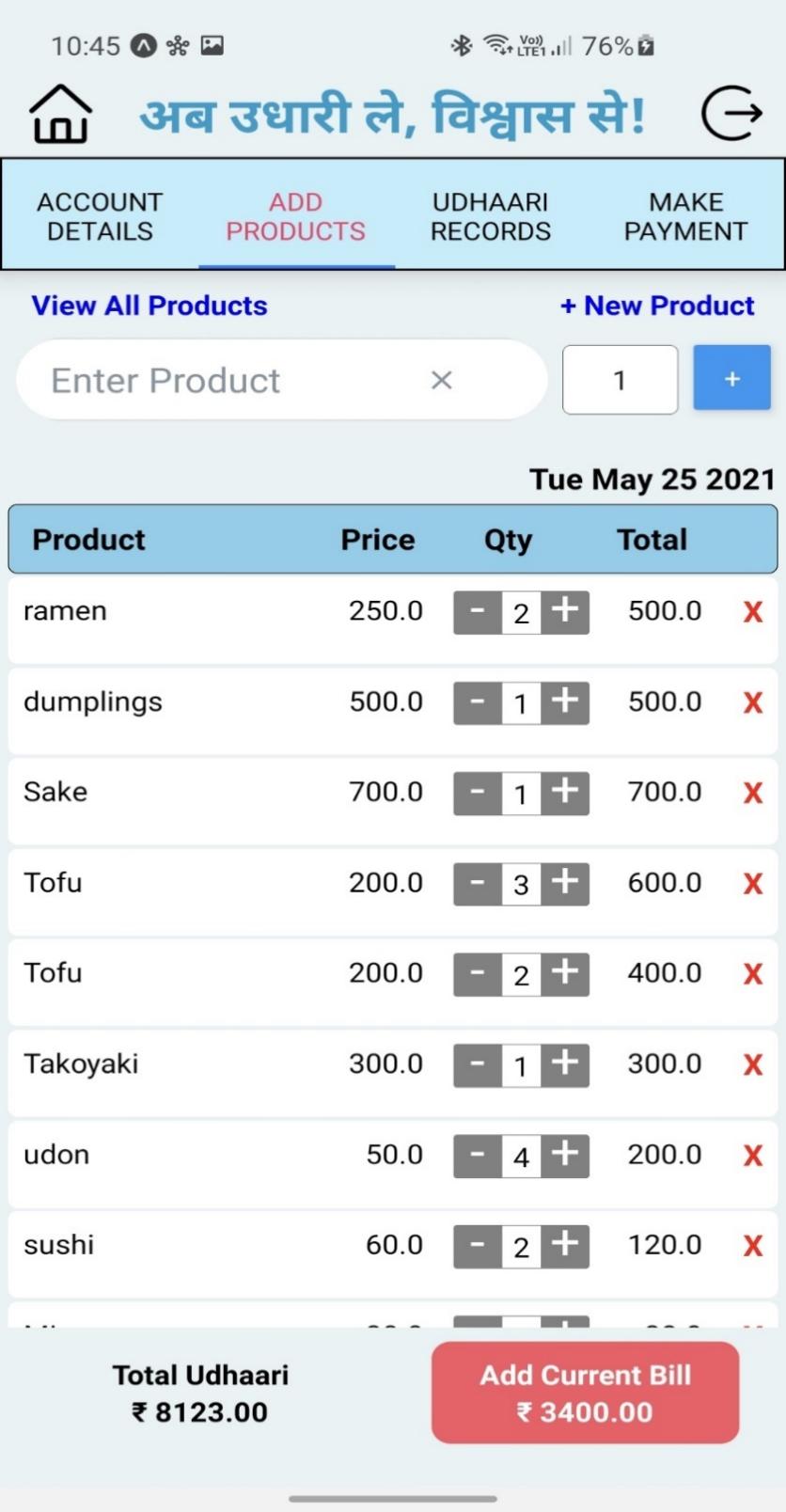
Add Products

* In this page vendors can search for products, add purchased products to purchase bills of selected consumer. Vendor (shopkeeper) needs to add product name, quantity, based on this base price and total price will be dynamically calculated. Along with this total purchase bill and total due amount is also displayed in this page.



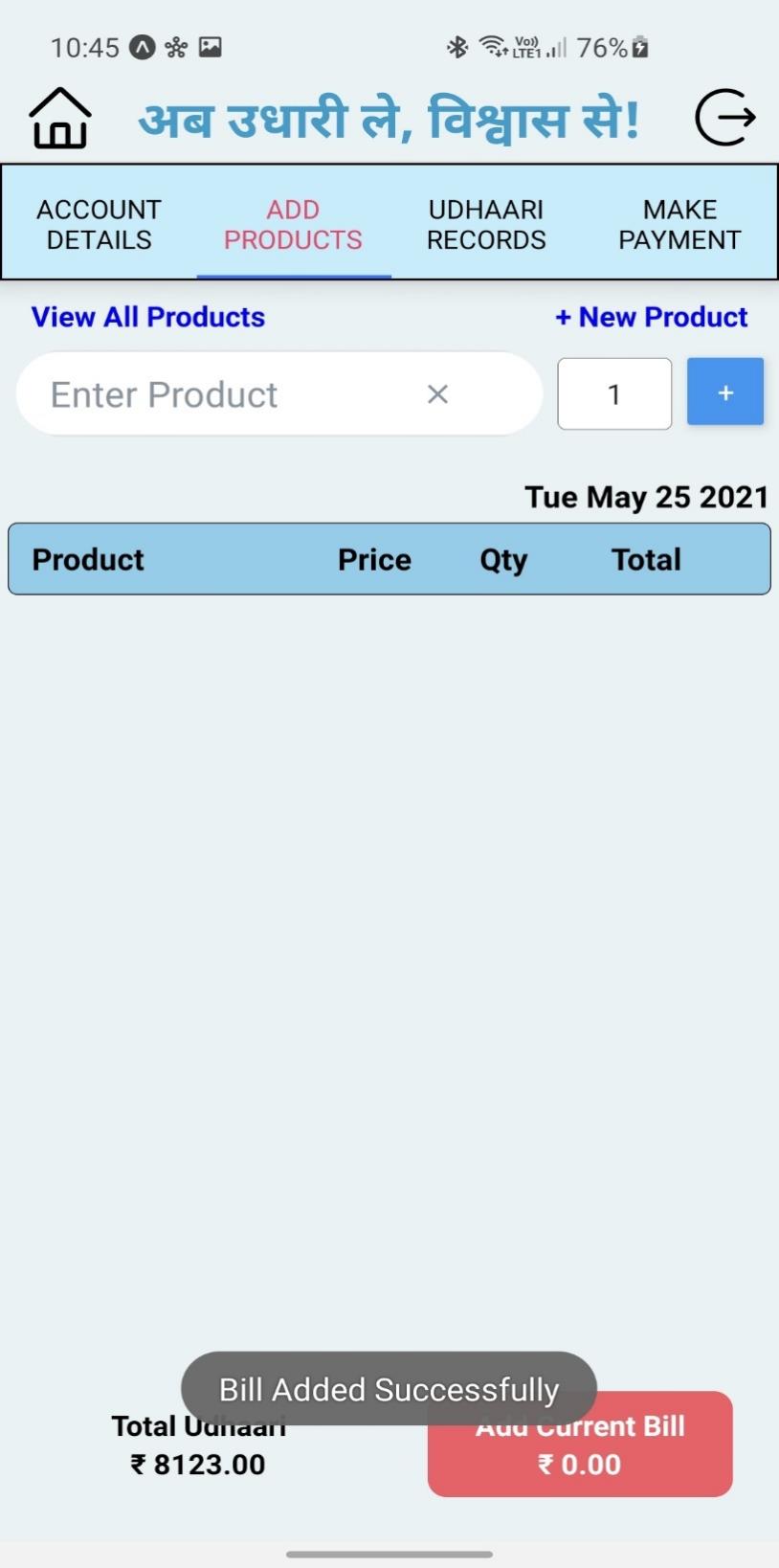
***Figure 7.14 Add Products***

* Vendors can add purchase products to the purchase list and also can delete any  product from the rows in the list. Add current bill amount and total udhaari amount will change dynamically based on whether product is added or deleted.



***Figure 7.15 Total Amount/Current Bill***

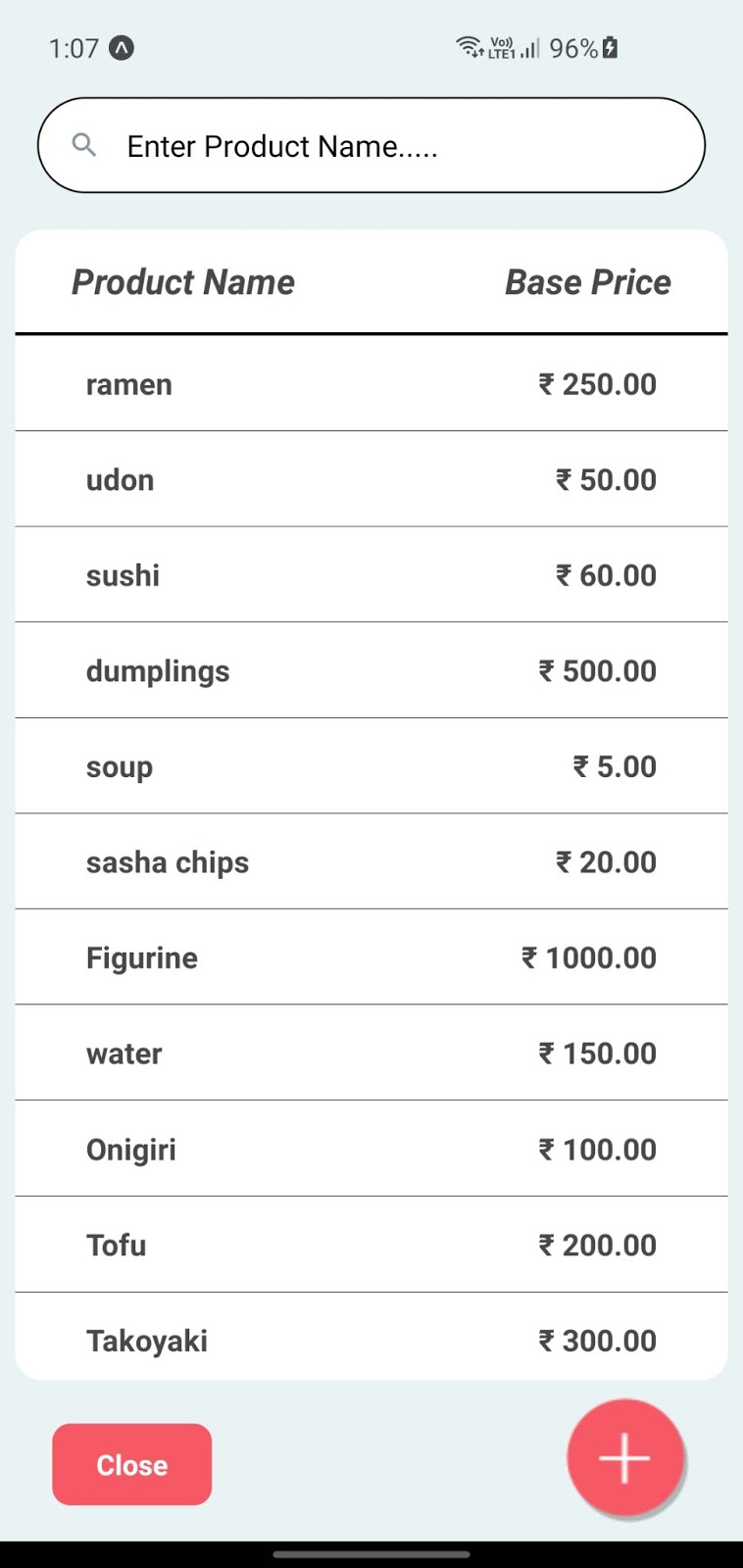
* When a vendor (shopkeeper) adds all purchase products and clicks on Add Current Bill, then current bill of that list of products is generated and saved in system.



***Figure 7.16 Bill Added Successfully***

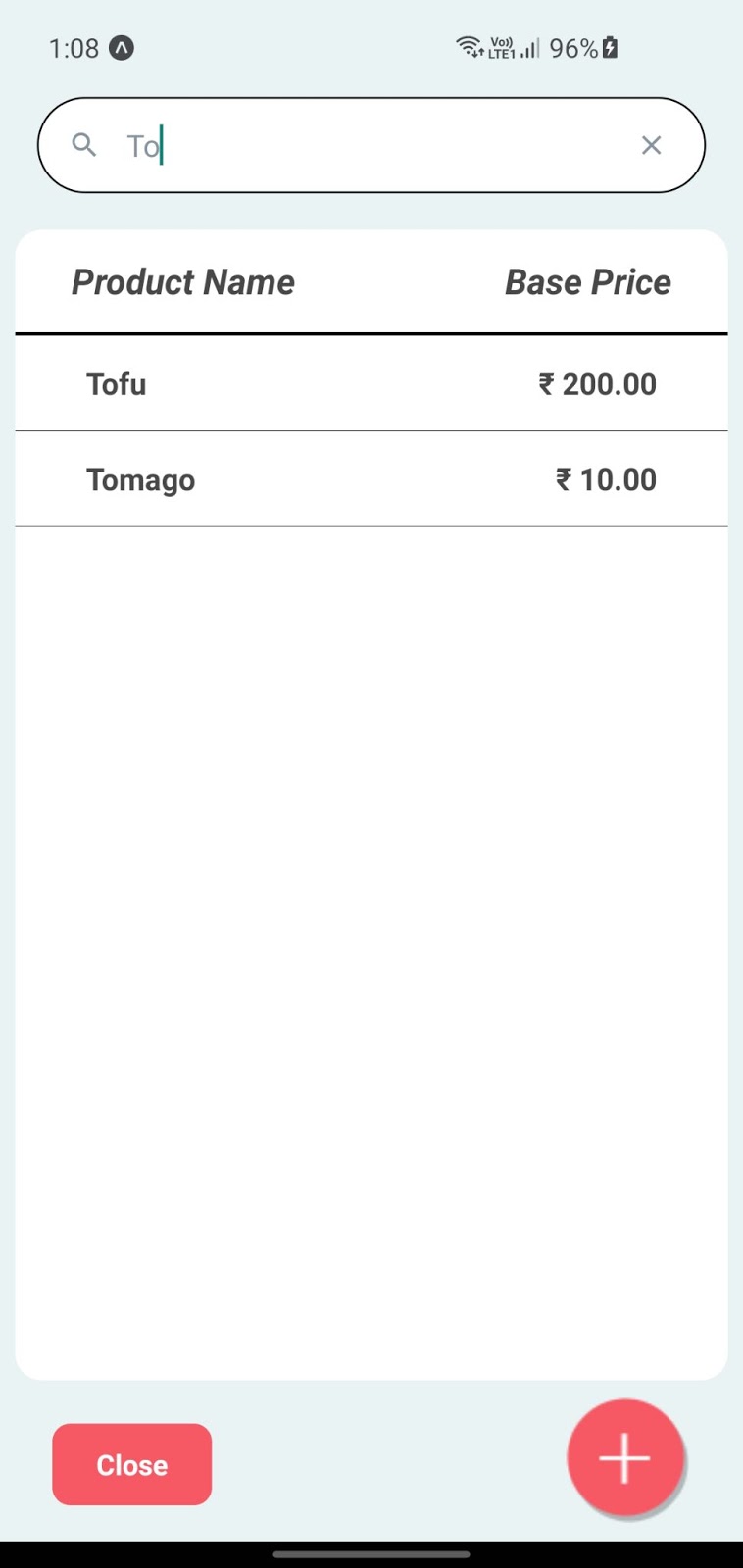
View All Products

* Vendor (shopkeeper) clicks on “View All Products”, then vendor can view all products available in the system along with their name and base price.



***Figure 7.17 View Products***

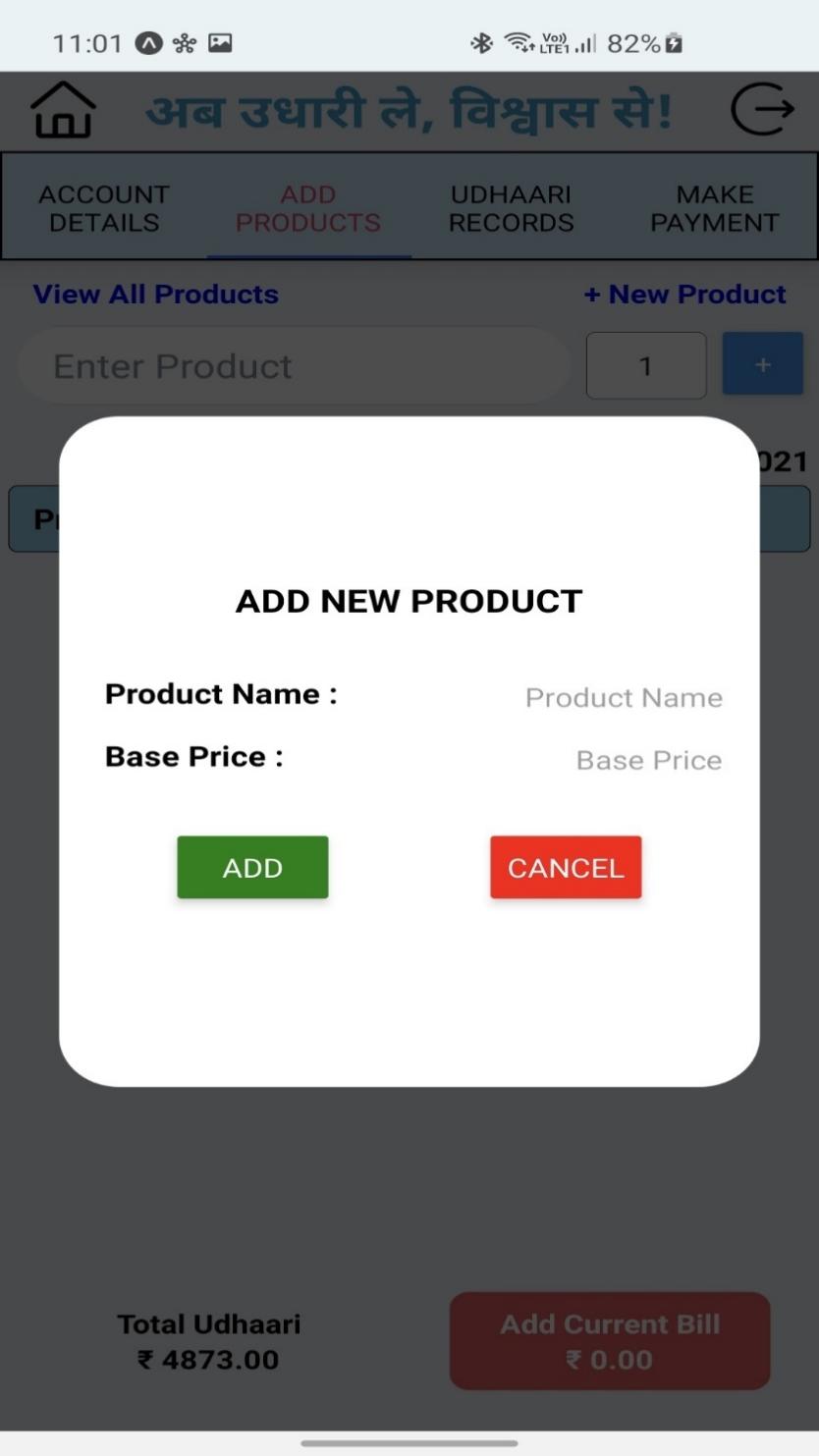
* Vendor (shopkeeper) can also search for particular product by using search bar



***Figure 7.18 Search A Product***

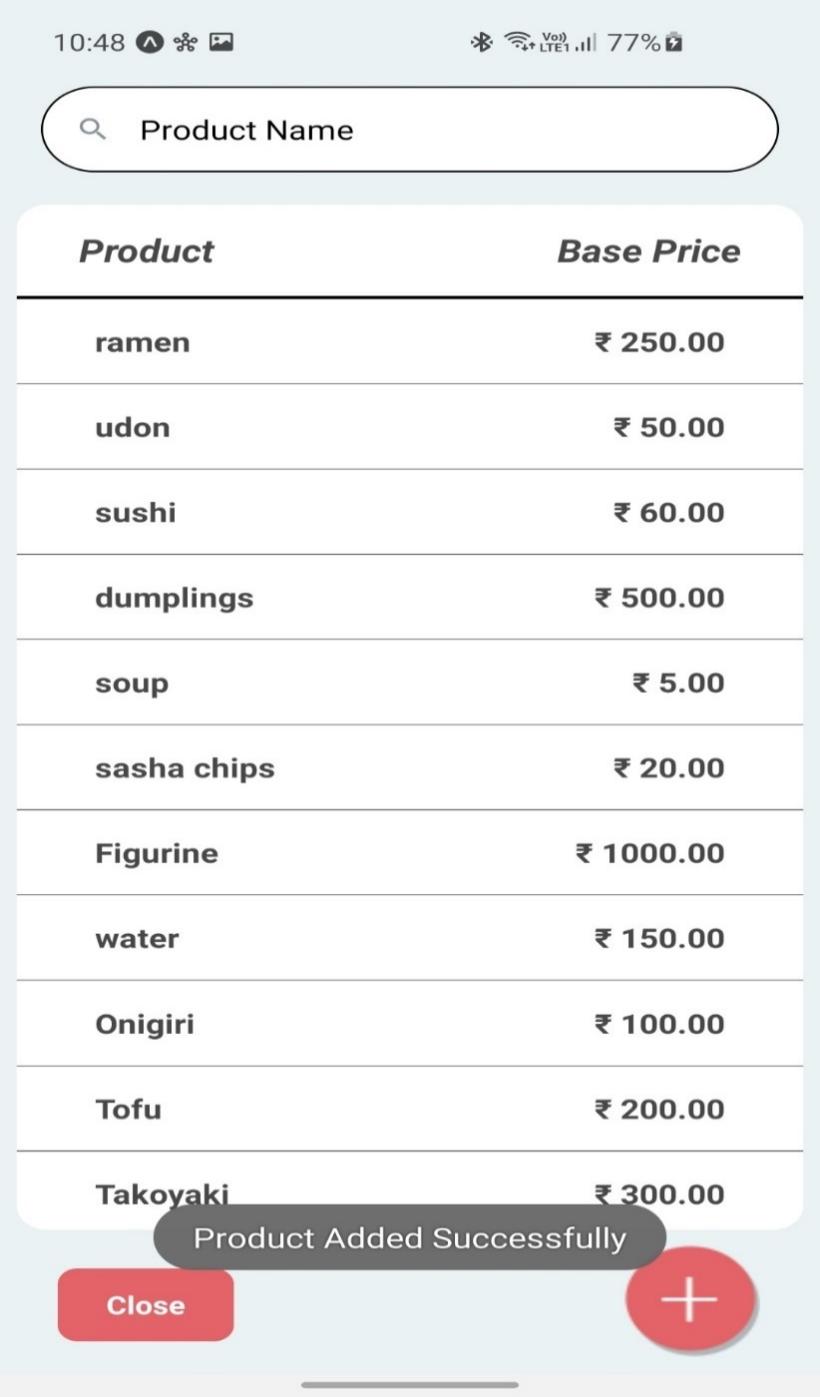
Add new Product

* Vendor (shopkeeper) clicks  on “Add new Product”, vendor can view a pop-up model to add new product with their name and base price. Vendor can also add new product by clicking on floating button in view all products.



***Figure 7.19 Add A New Product***

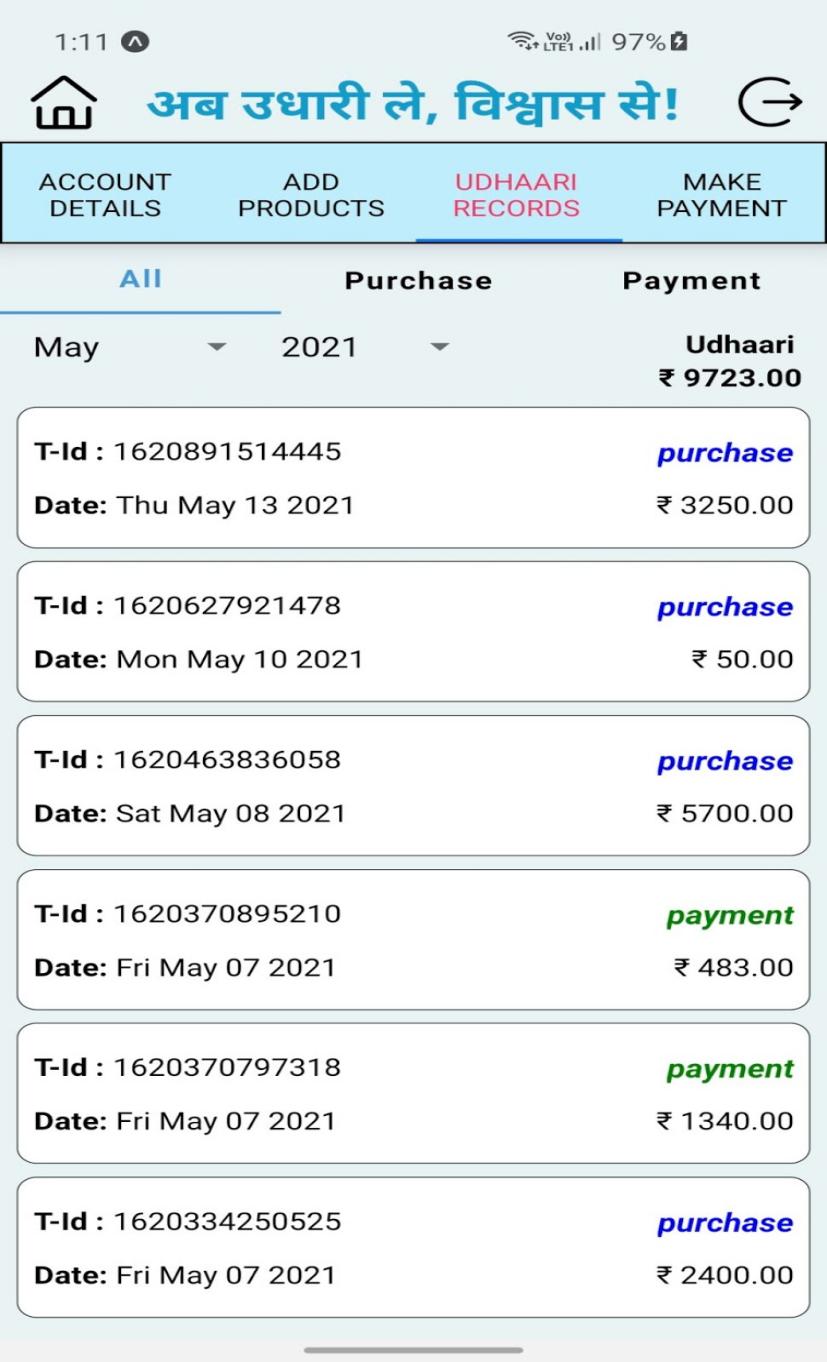
* Vendor (shopkeeper) adds new product and clicks on Add button, then system validates the values if all values are valid then system adds and displays that product to product list and redirects the vendor to view all products with message “Product Added Successfully”.



***Figure 7.20 Product Added Successfully***

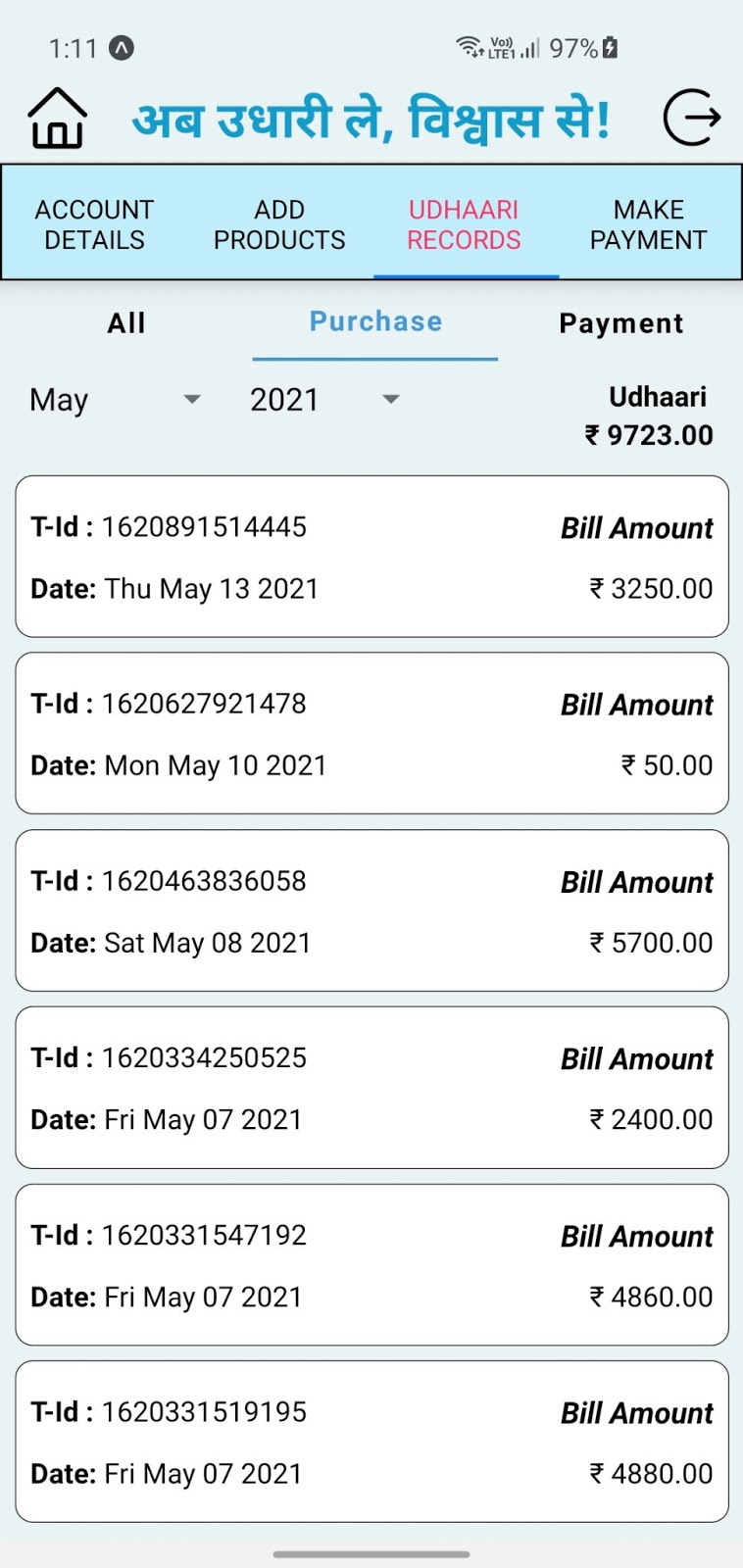
Udhaari Records

* In this page by default the vendor (shopkeeper) can view all transaction details of a selected consumer with their purchase bill as well as payment bill. By default the vendor can view transaction details of current month, but by selecting particular month and year vendor can view all transaction details of that month and year as well.



***Figure 7.21 Transaction History***

* On selecting a purchase, vendor (shopkeeper) can view all purchase bill for current month by default, by clicking on any card vendor can view all purchase products with all details of products. Vendor can also view purchase bills of other months by selecting particular month and year.



***Figure 7.22 Purchase Transaction***

* On selecting a payment, vendor (shopkeeper) can view all payment bill for current month by default, by clicking on any card vendor can view all payment details. Vendor can also view payment bills of other month by selecting particular month and year.



***Figure 7.23 Payment Transaction***

Purchase

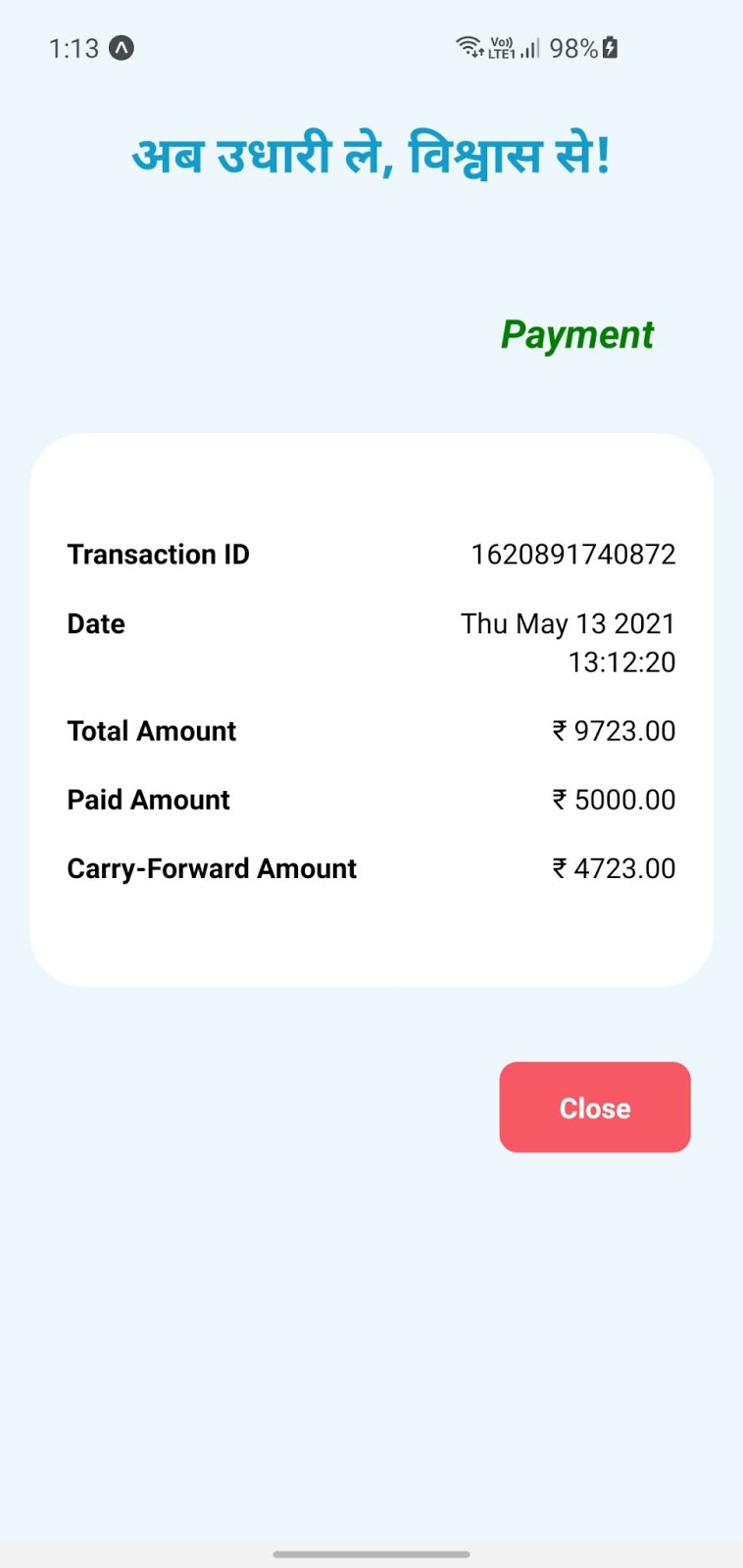
By clicking on any card of purchase bill, vendor (shopkeeper) can view all purchase details such as transaction\_id, date, purchased products, price, quantity and total bill.



***Figure 7.24 Purchase Bill***

Payment

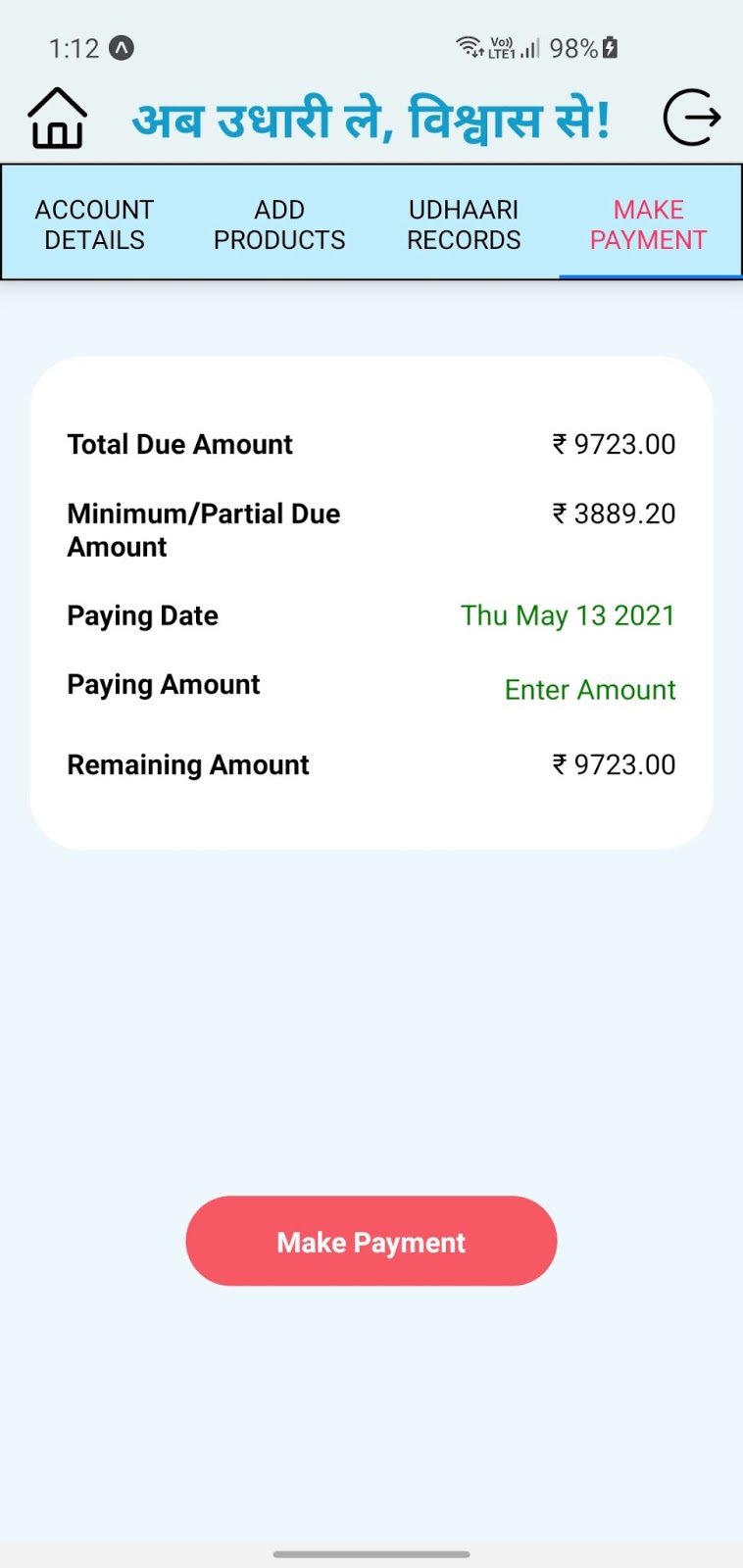
By clicking on any card of payment bill, vendor (shopkeeper) can view all payment details such as transaction\_id, date, total amount, paid amount and carry forwarded amount.



***Figure 7.25 Payment Bill***

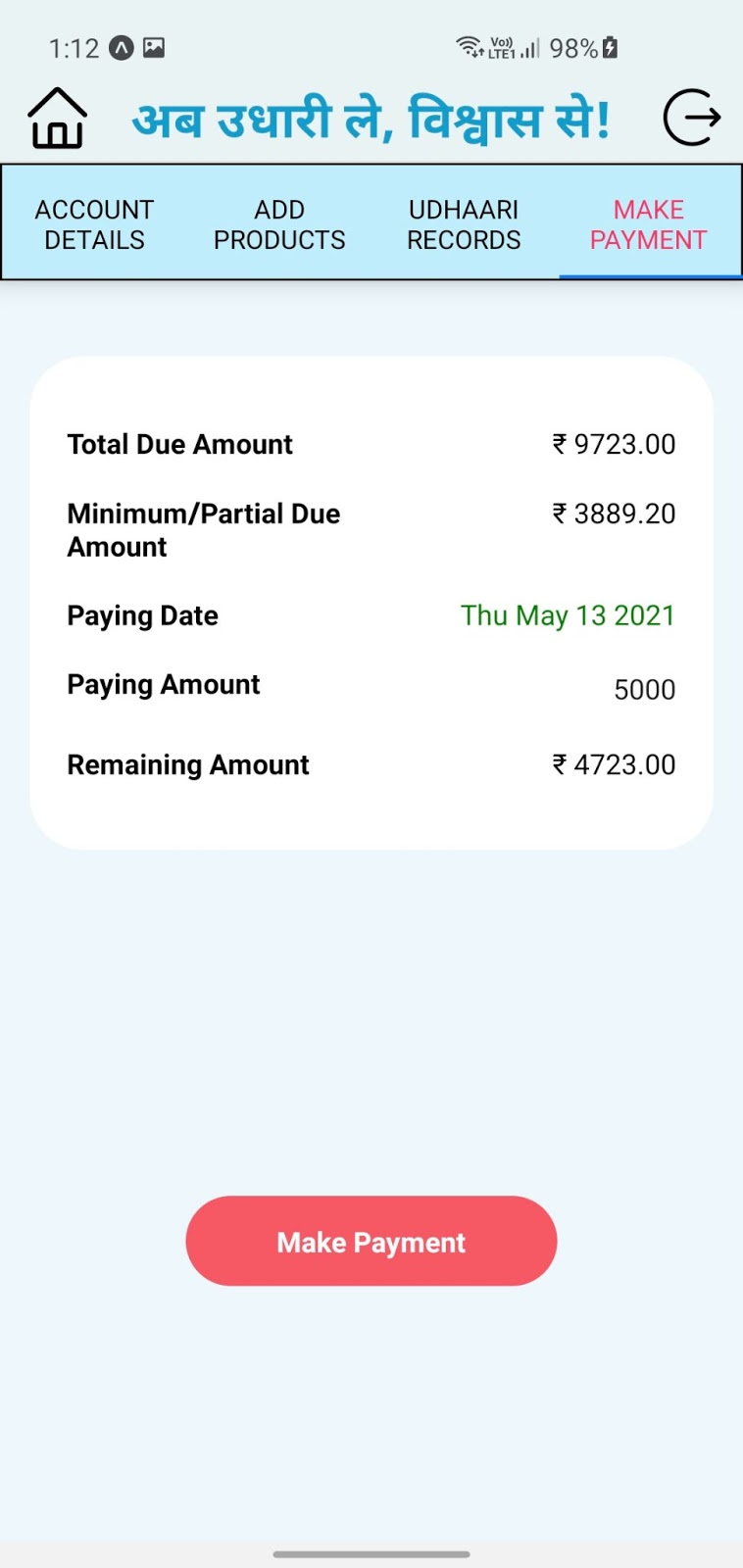
Make Payment

* On this page, vendors can make payments of selected consumers.



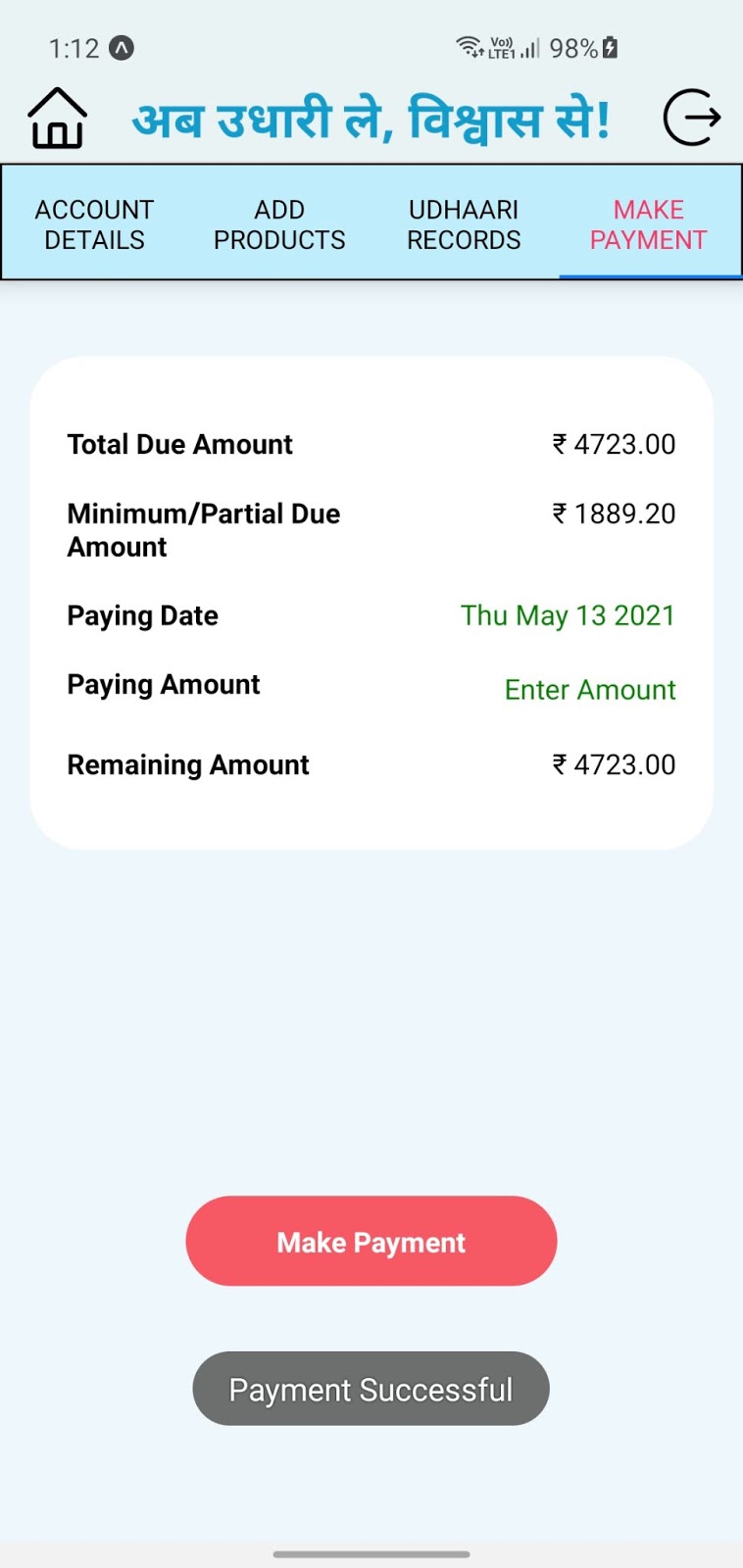
***Figure 7.26 Make Payment***

* Vendor (shopkeeper) needs to select paying date and paying amount, then system checks validation if paying amount is greater than or equal to threshold set by vendor. Afterwhich, system allows vendor to make payment transaction.



***Figure 7.27 Payment Details***

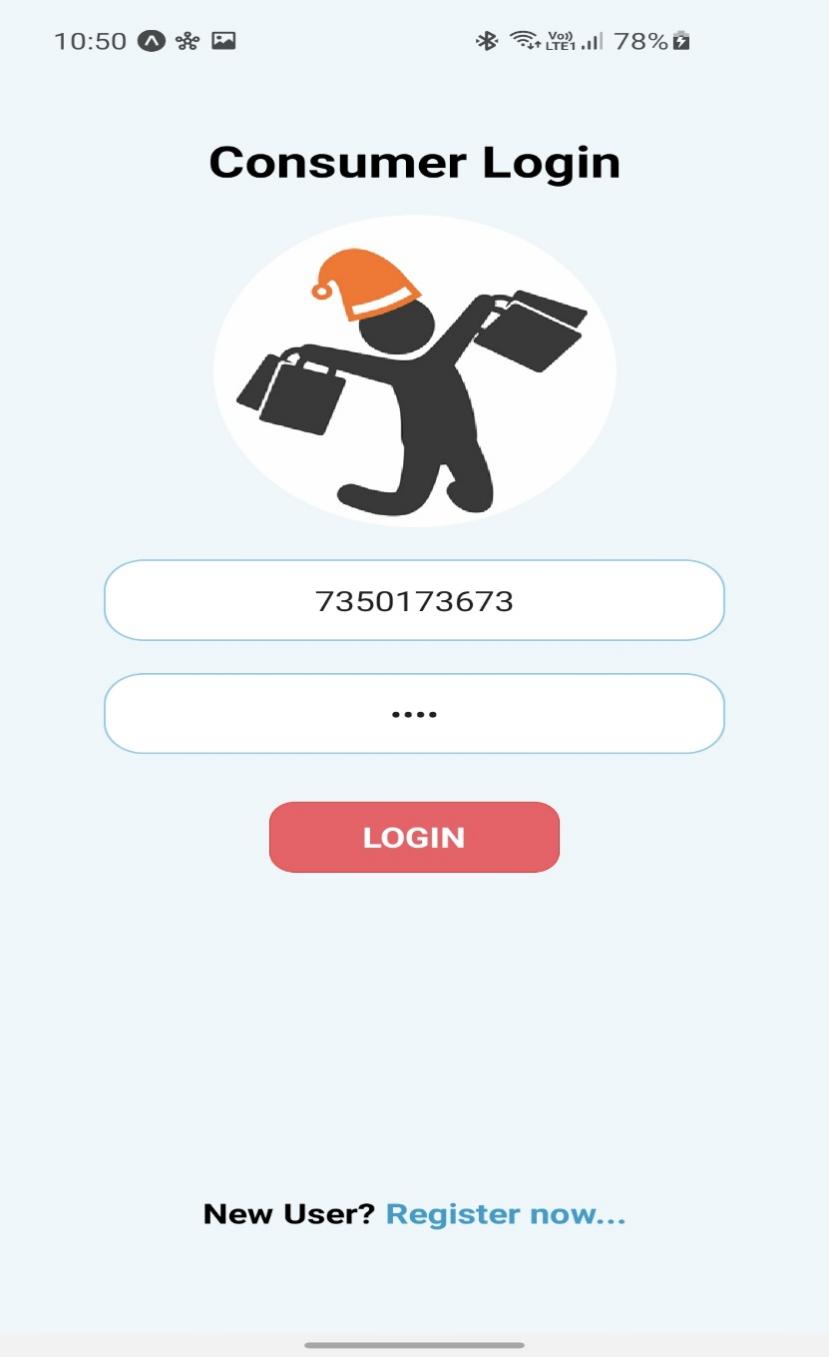
* If Vendor (shopkeeper) enters valid date and paying amount then system makes payment of selected amount and displays remaining amount on same page and adds payment bill in payment section and display the message “Payment Successful”.



***Figure 7.28 Payment Successful***

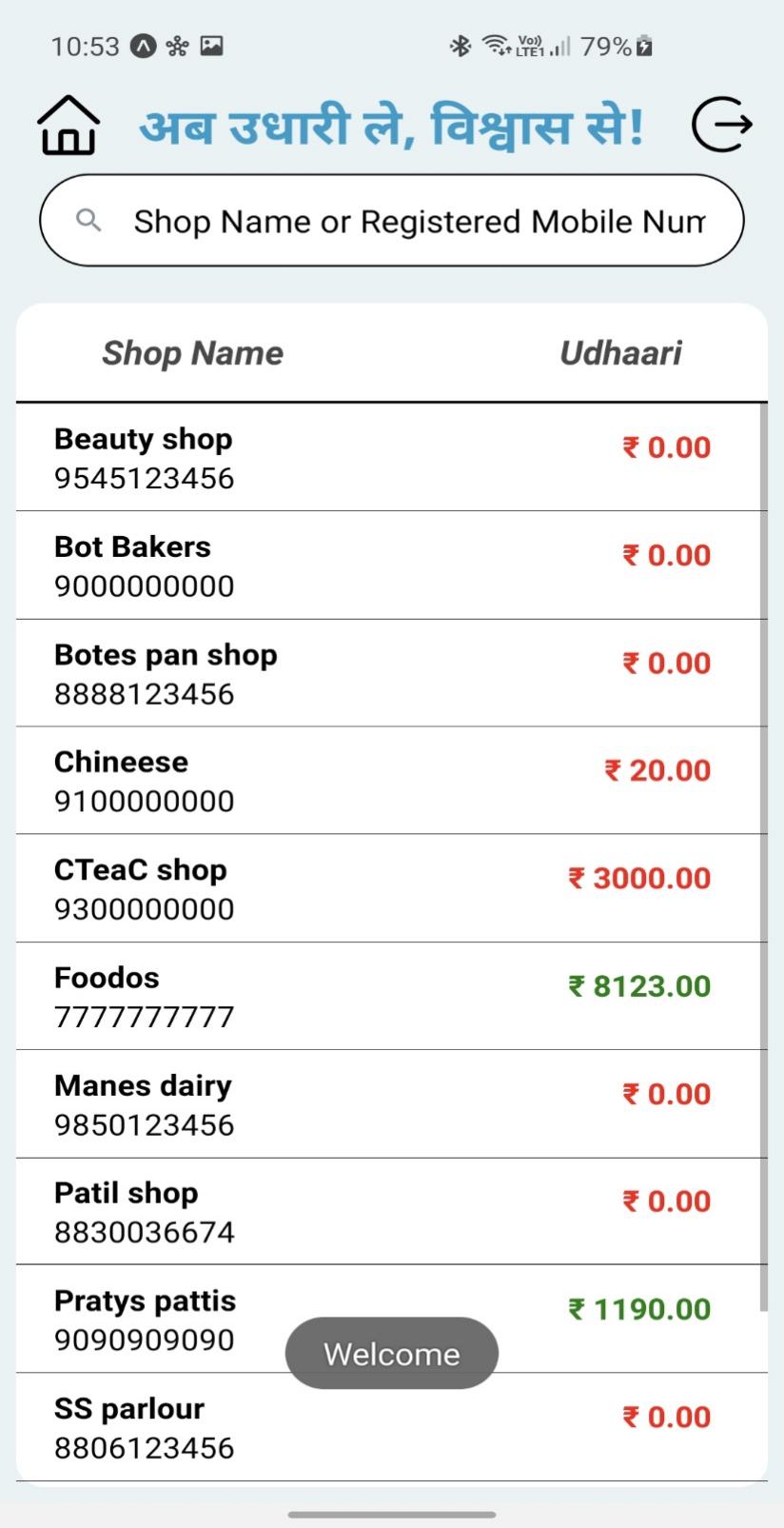
Consumer Login

* This is the login page of a consumer, if consumer is already registered then he/she can login directly with their login credentials.



***Figure 7.29 Consumer Login***

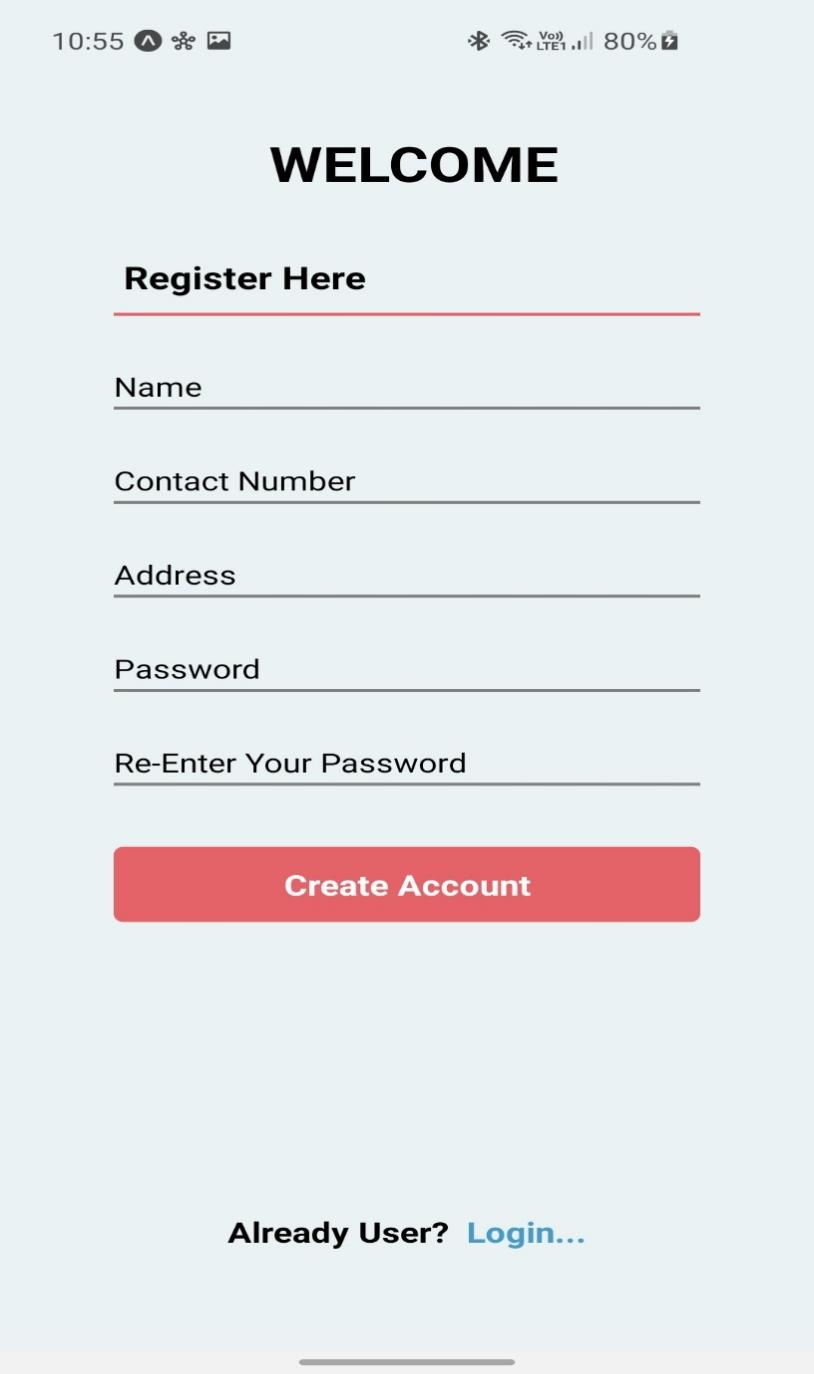
* System checks login credentials if contact number and password is valid then system display message “Welcome” and redirects the consumer to dashboard otherwise system provide validation error message.



***Figure 7.30 After Login Successful***

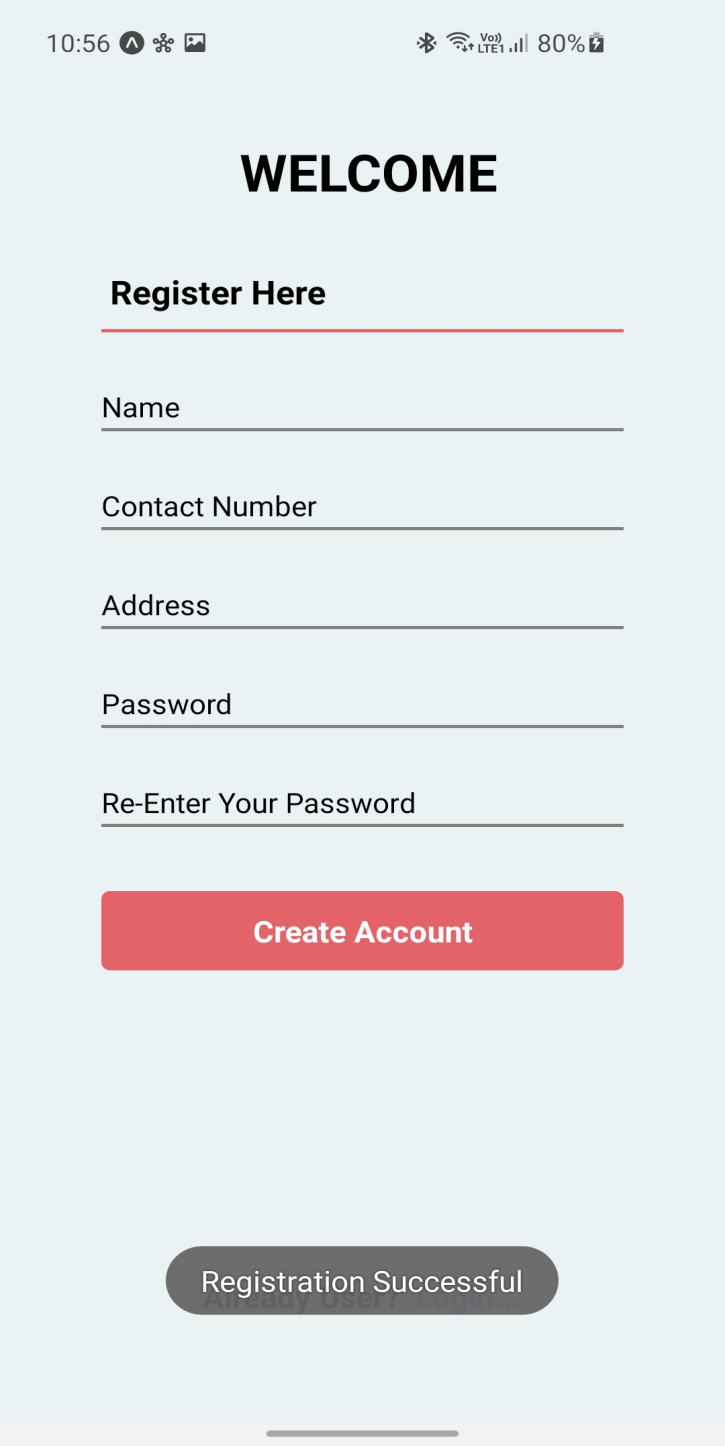
Consumer Register

* This is the Registration page for consumers. If a consumer is unregistered and wants to login then he/she has to register first by their name, contact number, address and password. System checks validations if all fields are correct system registers new vendor (shopkeeper) and allow vendor to login into the system.



***Figure 7.31 Consumer Registration***

* If consumer enter all valid credentials then system gives message of registration successful and allow consumer to login through their registered mobile number and password.



***Figure 7.32 Consumer Registration Successful***

***Consumer Dashboard***

This is a consumer dashboard, if consumer is successfully logged-in he/she will be redirected to this page. This is the homepage of the consumer's application. This page contains a search bar to search for a particular vendor (shopkeeper) using their shop name or registered mobile number with this there is a list of vendors with their shop name , registered mobile number and total due amount. Consumer can also logout from application by clicking on logout button.

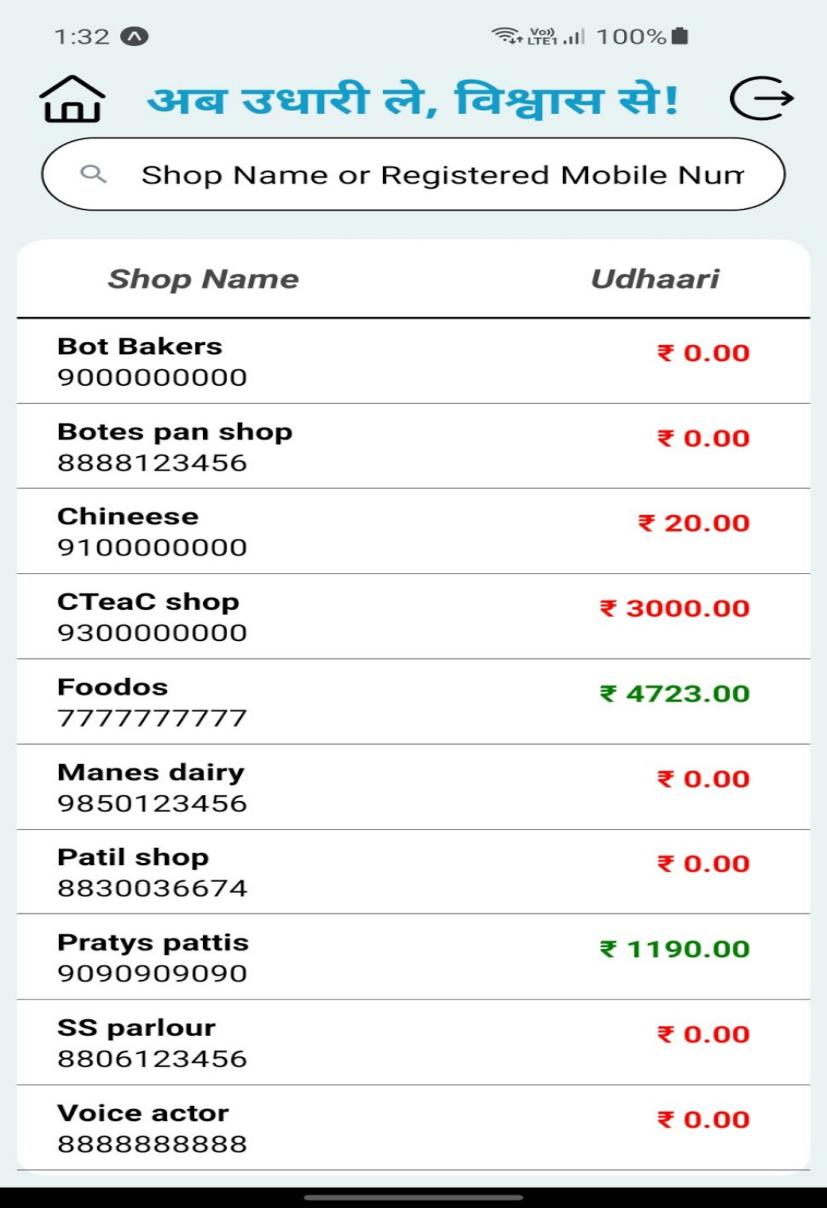


Figure 7.33 Consumer Dashboard

My Udhaari

* In this page by default the consumer can view all transaction details with their purchase bill as well as payment bill. By default the consumer can view transaction details of current month, and by selecting particular month and year consumer can view all transaction details of any month and year.

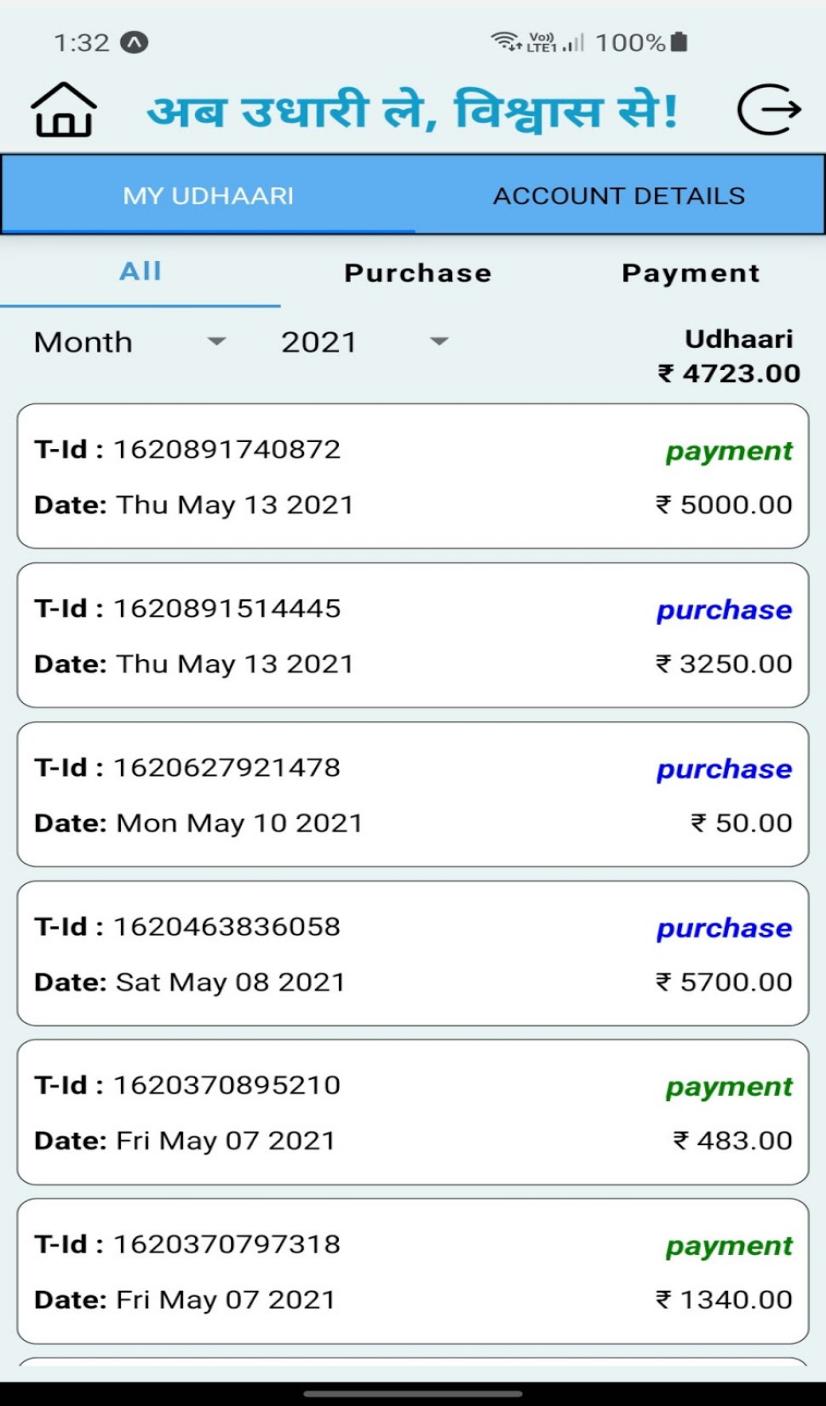


Figure 7.34 My Udhaari

* On selecting a purchase , consumer can view all purchase bill for current month by default, by clicking on any card consumer can view all purchase products with all details of products. Consumer can also view purchase bills of other month by selecting particular month and year.

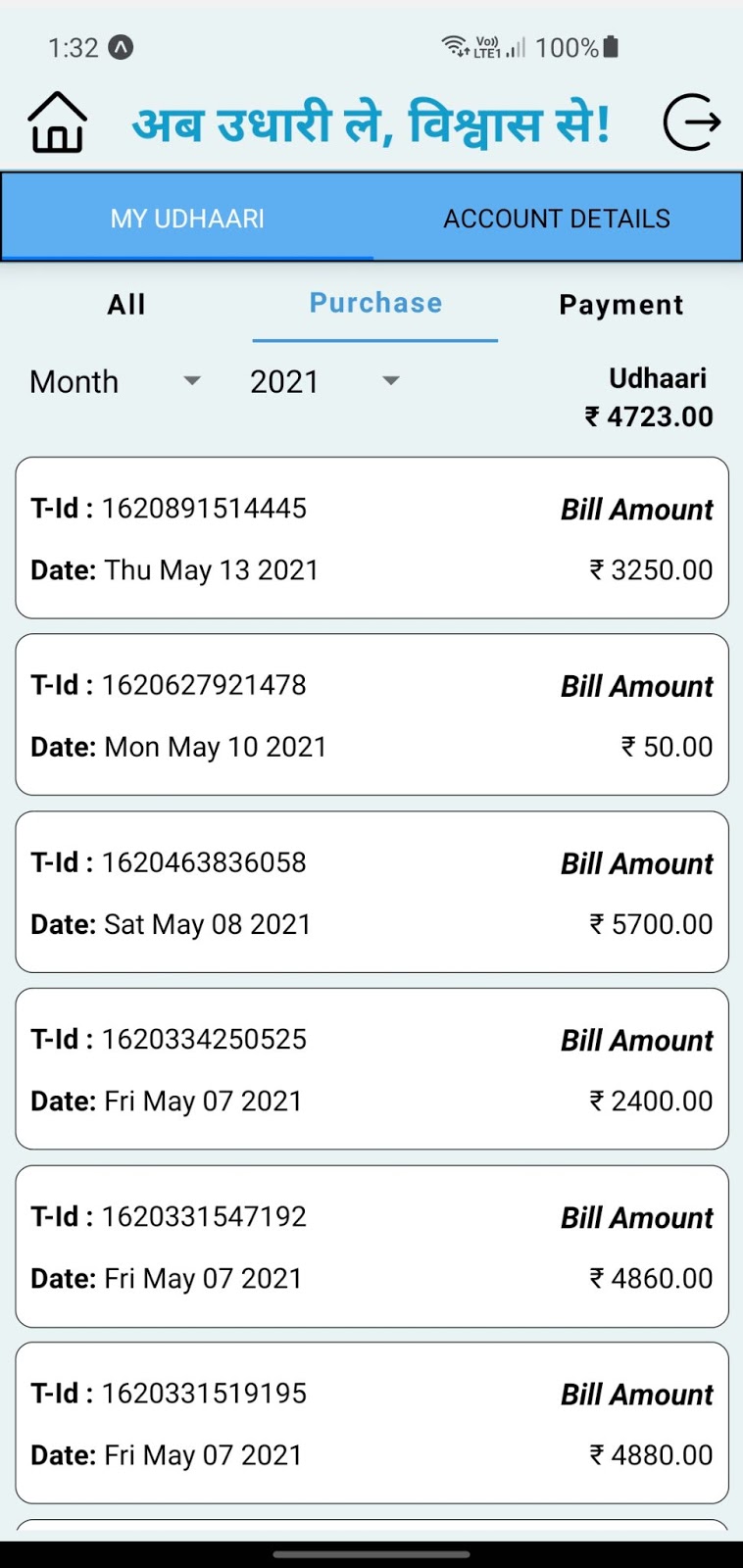


Figure 7.35 Purchase Transaction

* On selecting a payment, vendor (shopkeeper) can view all payment bill for current month by default, by clicking on any card consumer can view all payment details, consumer can also view payment bills of other month by selecting particular month and year.

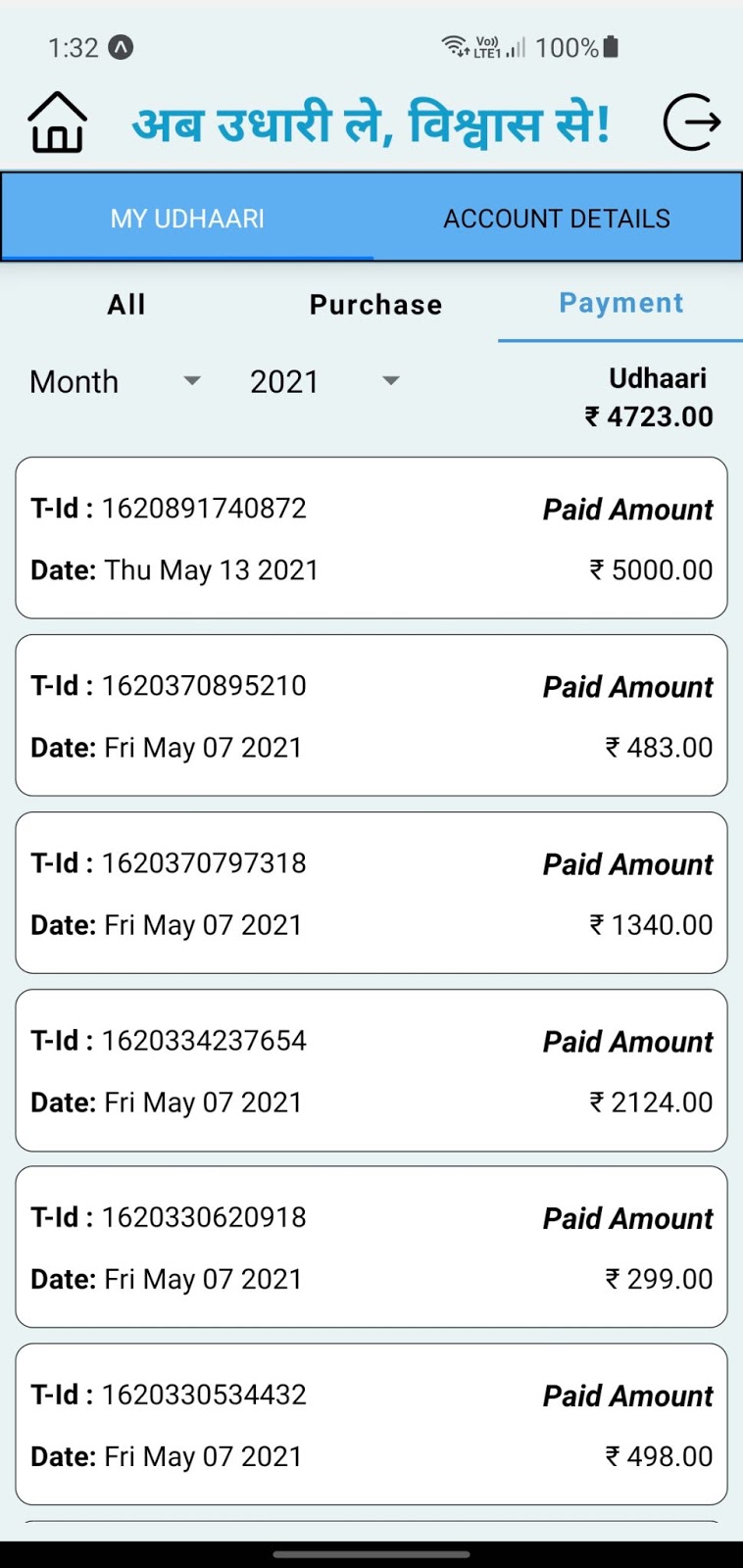


Figure 7.36 Payment Transaction

Purchase

By clicking on any card of purchase bill, consumer can view all purchase details such as transaction\_id, date, purchased products, price, quantity and total bill.



Figure 7.37 Purchase Bill

Payment

By clicking on any card of payment bill, consumer can view all payment details such as transaction\_id, date, total amount, paid amount and carry forwarded amount.

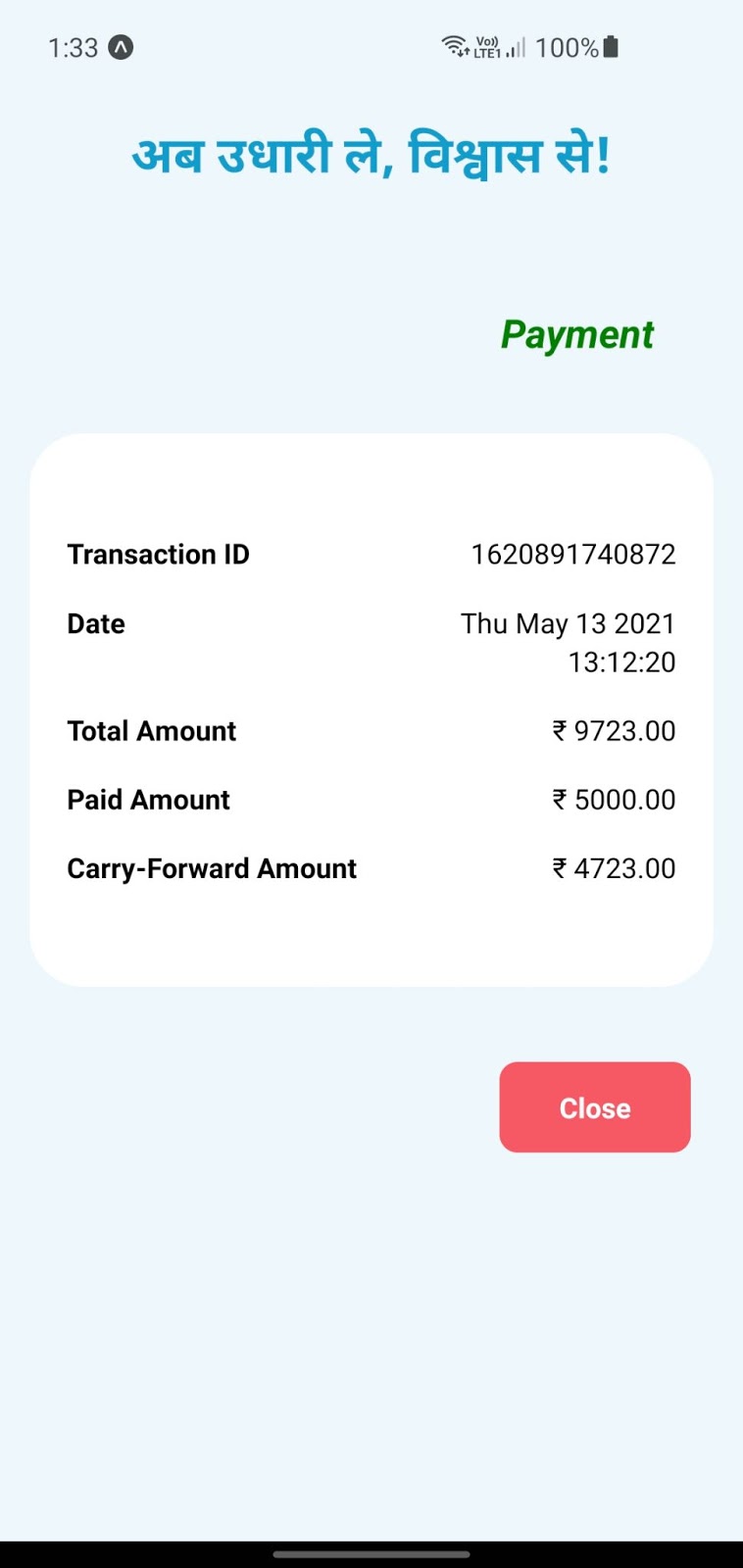


Figure 7.38 Payment Bill

Consumer Account Details

In this page consumer can view shop name, shop address and registered mobile number of vendor (shopkeeper) along with status of account. Consumer can also view  all details such as own account start date, billing start date, billing due date, last paid amount, total due amount and partial due amount.

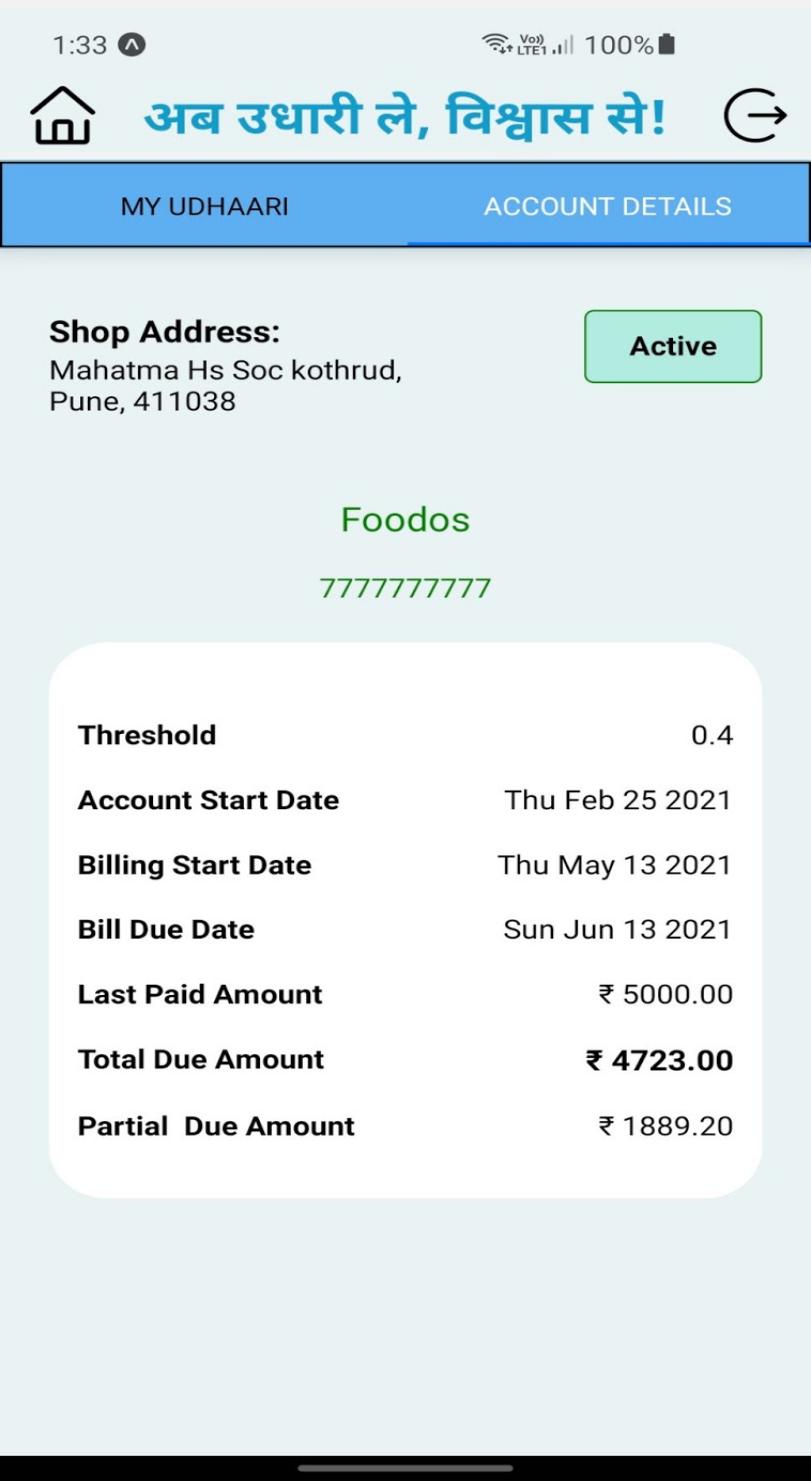


Figure 7.39 Consumer Account Details

Testing

This section is used to check whether the actual software product matches expected requirements and to ensure that software product is defect free. Here testing is performed  to identify errors, gaps or missing requirements in contrast to actual requirements.

Quality process

This is a process which describes, the quality of software products or services provided to the customers on improving the software development process and making it efficient and effective as per the quality standards defined for software products. UI testing and Functional testing of both frontend and backend API testing using chai and mochai is performed.

Test Cases identification

This section convey information about, the different test cases identified for each component of this application. They are as follows,

* Test Case  - UX01

Test Case Description: To verify that a Launching page is provided to select category as per the user’s role

Pre-condition: User opens the Udhaari Book application

Table 7.1 UI Testing – Launching Page

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | The first page rendered when the application starts is the Launching page containing two categories vendor (shopkeeper) and consumer | User selects the vendor category | Application must render the vendor (shopkeeper)’s Login page | Application renders the vendor (shopkeeper)’s Login page |
|  |  | User selects the consumer category | Application must render the consumer’s Login page | Application renders the consumer’s Login page |

* Test Case  - UX02

Test Case Description : To verify that a Registration page is provided for a vendor (shopkeeper) to register

Pre-condition : User selects vendor (shopkeeper) category and is a non-registered user

Table 7.2 Vendor Registration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | The first page rendered when the application starts is the Launching page containing two categories vendor (shopkeeper) and consumer. Selecting the vendor category renders the Login page.  Below there is a link to register for new users, User clicks New User? Register which opens the Registration page | Name, Contact Number, Shop Name, Shop Address, Set Password not entered | Application must alert the user about these missing fields | Application alerts the user about these missing fields stating,   1. Missing name - “Please Enter Your Name” 2. Missing contact number - “Please Enter Your Contact Number” 3. Missing shop name - “Please Enter Your Shop Name” 4. Missing shop address - “Please Enter Your Shop Address” 5. Missing password - “Please Enter Your Password” 6. Missing re-entry of password to verify it - “Please Re-Enter Your Password” |
|  |  | Invalid Contact Number | Application must alert the user with an error message | Application gives an alert stating, “Required 10 Digit Valid Contact Number” |
|  |  | Already Registered Mobile Number | Application must alert the user about the already  registered mobile number | Application gives an alert stating, “Contact Number Already Exists” |
|  | User’s contact number, password and other details are successfully stored in the database | Valid fields entered | Application must notify the user about successful registration | Application gives a notification stating, “Registration Successful” |

* Test Case  - UX03

Test Case Description : To verify that a Login page is provided for a vendor (shopkeeper) to login to Dashboard

Pre-condition : User selects vendor (shopkeeper) category and is a registered user

Table 7.3 Vendor Login

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | The first page rendered when the application starts after the user selects the vendor (shopkeeper) category is the Login page | RMN or/and Password not entered | Application must alert the user about these missing fields | Application alerts the user about these missing fields stating,   1. Missing RMN - “Please Enter Your Registered Mobile Number” 2. Missing password - “Please Enter Your Password” 3. Missing RMN and password - “Please Enter RMN And Password” |
|  |  | Unregistered contact number | Application must alert the user about contact number unregistered | Application gives an alert stating, “User Not Found! Please Register First To Login” |
|  |  | Invalid RMN/password | Application must alert the user with an error message | Application alerts the user about these invalid fields stating,   1. Invalid RMN - “Required 10 Digit Valid Contact Number” 2. Invalid Password - “Please Enter A Valid Password” |
|  | User successfully signs in and is redirected to the Dashboard | Valid fields entered | Application must notify the user about successful login | Application gives a message stating, “Welcome” |

* Test Case  - UX04

Test Case Description : To verify that a Dashboard UI is provided to the vendor (shopkeeper)

Pre-condition : User is a successfully logged in vendor (shopkeeper) and is on the Dashboard which is also a default landing page after successful login

Table 7.4 Vendor Dashboard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | The first page rendered when the application starts after user logins the vendor (shopkeeper) category is the Dashboard | User is rendered on Dashboard | Application must display list of all consumers with their name, RMN and total due amount, search bar to search a consumer, add floating button and logout button | Application displays a list of all registered consumers with their name, RMN and total due amount. A search bar to search a consumer using name or contact, a add floating button to add a new consumer and home and logout button |
|  |  | User selects a consumer from the list | Application must  redirect to the Account Details page for that selected consumer | Application successfully renders the Account Details page for selected consumer |
|  |  | User clicks the search bar | Application must show a pop-up keypad on the screen to type the credentials of a consumer | Application shows keypad pop-up on the screen to type the consumer credentials and based on the search filters the consumer list |
|  |  | User clicks the Add floating button | Application must allow to add a new consumer | Application successfully renders the Add Consumer page to add a new consumer |
|  | After clicking on any option on the Dashboard the user must be able to perform required action | User clicks the Logout button | Application must  allow user to logout successfully and redirects to the Login page | Application successfully allows user to Logout of the application and redirects to the Login page |

* Test Case  - UX05

Test Case Description : To verify that a search bar is provided for a vendor (shopkeeper) to search a consumer

Pre-condition : User is a valid logged-in vendor (shopkeeper) and is on Dashboard of the application

Table 7.5 Vendor Dashboard – Search Bar

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | To select a specific consumer among all the consumers using a search bar | Invalid RMN or consumer’s name | Application will show message for invalid input without any search results | Application gives blank consumer list without any search results |
|  |  | Valid consumer RMN | Application will display that consumer with name | Application will display that consumer with name, RMN and total due amount highlighted in red (blocked account) and green (active account) |
|  | User selects the consumer and successfully gets redirected to that consumer’s Account Details page | Valid consumer name | Application will display list of all consumer/s of searched name with their RMN | Application will display that consumer with name, RMN and total due amount highlighted in red (blocked account) and green (active account) |

* Test Case  - UX06

Test Case Description : To verify that a form to add a new consumer is provided to the vendor (shopkeeper)

Pre-condition : User is a valid logged-in vendor (shopkeeper) and clicks on the add floating button

Table 7.6 Add New Consumer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | To add a new consumer to the consumers list, first the vendor (shopkeeper) must search consumers by their RMN and then other details get autogenerated for the registered consumer  User clicks the Add consumer button | Contact Number, Threshold, Start date not entered | Application must alert the user about these missing fields | Application alerts the user about these missing fields stating,   1. All fields missing - “Please Search A Consumer” 2. Missing Threshold -  “Please Enter A Threshold” 3. Missing Start Date - “Please Select A Start date” |
|  |  | Invalid Contact number | Application must alert the user with an error message | Application shows no output |
|  |  | Already Registered Mobile Number | Application must alert the user about the already registered mobile number | Application gives an alert stating, “Consumer Already Exists” |
|  |  | Invalid Threshold | Application must alert user about invalid threshold | Application gives an alert stating, “Please Enter A Valid Threshold! Threshold must be a value between 0-1” |
|  |  | Threshold out of range | Application must alert user about threshold being out of range | Application gives an alert stating, “Threshold must be a value between 0-1” |
|  | Consumer’s name is stored in vendor’s consumer list | Valid fields entered | Application must notify the vendor and adds the new consumer to list | Application gives a notification stating, “Consumer Added Successfully” |

* Test Case  - UX07

Test Case Description : To verify that UI is provided to vendor (shopkeeper) for a selected consumer with options to navigate

Pre-condition : User is logged in vendor (shopkeeper) and has selected a consumer

Table 7.7 Vendor – Navigation Tab

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | The first page rendered when the user selects any consumer is the Account Details page of that consumer | User clicks Account Details | Application must  direct to the Account Details page containing consumer’s name, RMN, threshold, start date and due date | Application successfully directs to the Account Details page containing selected consumer’s name, RMN, address, threshold, account status, threshold, account start date, billing start date, bill due date, last paid amount, total due amount, partial due amount with save changes and delete button |
|  |  | User clicks Add Products | Application must  direct to Add Product page | Application successfully directs to the Add Product’s page |
|  |  | User clicks Udhaari Records | Application must  direct to the Udhaari Records page | Application successfully directs to the Udhaari Record’s page having all purchase and payment transactions of selected consumer |
|  |  | User clicks Make Payment | Application must direct to the Make Payment page | Application successfully directs to the Make Payment’s page |
|  |  | User clicks Back to Dashboard icon | Application must  direct to the Dashboard where list of all consumers is available | Application successfully directs to the Dashboard consisting of all consumers list |
|  | After clicking on any nav option on the screen the user must be able to go to that directed page | User clicks  Logout | Application must  allow user to logout successfully and redirects to the Login page | Application successfully allows user to Logout of the application and redirects to the Login page |

* Test Case  - UX08

Test Case Description : To verify that an edit button is provided to edit account fields of a selected consumer

Pre-condition : User is a successfully logged-in vendor (shopkeeper) and is on Account Details page of a selected consumer

Table 7.8 Edit Account

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User views all account details and wants to edit name or threshold fields.  The editable fields are highlighted | Name and threshold not changed | Application must alert the user about these missing fields | Application gives an alert stating, “You have not made any changes” |
|  |  | Name not entered | Application must alert the user with an error message | Application gives an alert stating, “Please Enter A Name” |
|  |  | Invalid Name | Application must alert the user with an error message | Application gives an alert stating, “Please Enter A Valid Name” |
|  |  | Valid Name | Application must notify the user about successful changes | Application gives a notification stating, “Name Changed Successfully” |
|  |  | Threshold not entered | Application must alert the user with an error message | Application gives an alert stating, “Please Enter A Threshold” |
|  |  | Invalid Threshold | Application must alert the user with an error message | Application gives an alert stating, “Please Enter A Valid Threshold” |
|  |  | Threshold out of range | Application must alert the user with an error message | Application gives an alert stating, “Threshold must be a value between 0-1” |
|  | The old fields must be replaced with the changed one’s and the new data must be stored in the database | Valid Threshold | Application must notify the user about successful changes | Application gives a notification stating, “Threshold Changed Successfully” |

* Test Case  - UX09

Test Case Description : To verify that a delete option is provided to delete an account of a selected consumer

Pre-condition : User is a successfully logged-in vendor (shopkeeper) and is on Account Details page of a selected consumer

Table 7.9 Delete Account

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User clicks the delete icon for closing the account    Consumer’s name and RMN deleted from the consumers list | On clicking the delete icon confirmation message is provided | Application must ask the user for confirmation of deleting account | Application asks the user to confirm delete,   1. On clicking No - user remains on the same page 2. On clicking Yes - Application gives a successful notification stating, “Account Deleted Successfully” and user is redirected to the dashboard |

* Test Case  - UX10

Test Case Description : To verify that a table is provided to enter product details purchased for a selected consumer

Pre-condition : User is a successfully logged-in vendor (shopkeeper) and is on Add Products page of a selected consumer

Table 7.10 Add Products Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User searches required product, if product is available it is displayed with its base price | Product not entered | Application must alert the user about these missing fields | Application gives an alert stating, “Please Enter the product” |
|  |  | Quantity not entered | Application must alert the user  about it | Application by default takes quantity to be 1 |
|  |  | Invalid Quantity entered | Application must alert the user about invalid quantity | Application gives an alert stating, “Please Enter A Valid Quantity” |
|  |  | User clicks add button | Application must add the product to the table | Application adds the product to the table as an udhaari and auto calculates the total price as the product of quantity and base price |
|  |  | User wishes to increase or decrease the quantity after adding the product to the table | Application must allow user to add or decrease the quantity after adding the product to the table | Application allows user to increase decrease the quantity by providing increment or decrement button for each quantity |
|  | All product details are stored in the database | User wishes to remove a product from the table | Application must allow user to remove the product from table | Application provides a remove icon on each row of the table through which any product added to the table can be removed |

* Test Case  - UX11

Test Case Description : To verify that a dynamic view is provided to display the total amount after adding a product

Pre-condition : User is on Add Products page of a selected consumer and successfully has added details of a purchased product

Table 7.11 Total Amount/ Current Bill

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User enters the product, quantity and base price from which the total price is successfully calculated | User clicks the add button to add each product | Application must calculate the current total amount as well as the total udhaari amount for each added product and display it dynamically | Application calculates and displays both the amount which changes on addition of each product to the table |
|  |  | User does not click the add current bill button | Application must display the current bill to be 0 itself | Application does not add any amount to the current bill displaying it to be zero and also with no change in total Udhaari |
|  |  | User clicks the add current bill button | Application must add the product to purchase history | Application adds the current bill successfully to the purchase history |

* Test Case  - UX12

Test Case Description : To verify that a link is provided to view all the available products with their base price

Pre-condition : User is a successfully logged-in vendor (shopkeeper) and is on Add Products page of a selected consumer

Table 7.12 View Products

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User clicks the View All Products link to view all the available products | On clicking View All Products | Application must display all available products with their base price | Application displays list of all available products with their base price, a search bar to search any product and a floating button to add a new product if required |

* Test Case  - UX13

Test Case Description : To verify that a link is provided to add a new product with its base price

Pre-condition : User is a successfully logged-in vendor (shopkeeper) and is on Add Products page of a selected consumer

Table 7.13 Add A New Product

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User clicks the + New Product to add a new product with base price | On clicking  + New Product | Application must display a modal through which new product with its base price can be added | Application displays a modal to add a new product with its base price if required product with its base price is not available in the products list |

Test Case  - UX14

Test Case Description : To verify that the records of all transactions history is provided to the vendor (shopkeeper)

Pre-condition : User is a successfully logged-in vendor (shopkeeper) and is on Udhaari Records page of a selected consumer

Table 7.14 Udhaari Records – Transaction History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User has options to see all transactions as well as separate payment and purchase transaction history | User does not click any option | Application must by default show all the latest records of transaction  history | Application shows all latest payment and purchase history transaction by default |
|  |  | User clicks on any one transaction | Application must show transaction description in details | Application shows detailed description of that transaction in a bill format |
|  |  | User filters the active years to see the payment history | Application must show all the transactions based on the filtered year and month | Application shows all purchase transaction based on the filtered year and month |
|  |  | User filters an inactive year or invalid year  to see the payment history | Application must notify the user about it | Application shows no records as the years were not active |

* Test Case  - UX15

Test Case Description : To verify that the records of purchase history is provided to the vendor (shopkeeper) of selected consumer

Pre-condition : User is a successfully logged-in vendor (shopkeeper) and is on Udhaari Records page of a selected consumer

Table 7.15 Purchase Transaction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User selects payment history with filter for year and month | User does not click purchase history | Application must by default show the latest records of all payment as well as purchase history | Application shows all the latest transactions of both payment as well as purchase history |
|  |  | User clicks purchase history | Application must show all the records of purchase transactions | Application shows all purchase transactions with latest transaction as the first one |
|  |  | User filters the year and month to see the purchase history accordingly | Application must show all the transactions based on the filtered year and month | Application shows all purchase transaction based on the filtered year and month |
|  |  | User clicks on one of the transactions | Application must show details about that transaction | Application shows detailed description of that transaction in a bill format |
|  |  | User filters an inactive month/year or invalid month/year  to see the purchase history | Application must notify the user about it | Application shows no records as the years were not active |

* Test Case  - UX16

Test Case Description : To verify that the records of all payment history is provided to the vendor (shopkeeper)

Pre-condition : User is a successfully logged-in vendor (shopkeeper) and is on Udhaari Records page of a selected consumer

Table 7.16 Payment Transaction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User selects payment history with filter for year and month | User does not click payment history | Application must by default show the latest records of all payment as well as purchase history | Application shows all the latest transactions of both payment as well as purchase history |
|  |  | User clicks payment history | Application must show all the records of payment transaction | Application shows all payment transaction with latest transaction as the first one |
|  |  | User filters the active years and months to see the payment history | Application must show all the transactions based on the filtered year and month | Application shows all payment transaction based on the filtered year and month |
|  |  | User clicks on one of the transactions | Application must show details about that transaction | Application shows detailed description of that transaction in a bill format |
|  |  | User filters an inactive year or invalid year  to see the payment history | Application must notify the user about it | Application shows no records as the years were not active |

* Test Case  - UX17

Test Case Description : To verify that the Make Payment option is provided to the vendor (shopkeeper) for selected consumer

Pre-condition : User is a successfully logged-in vendor (shopkeeper) and is on Make Payment page of a selected consumer

Table 7.17 Make Payment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | Make payment page is rendered when the user clicks on the make payment option in navbar | Currently paying amount not entered | Application must alert the user about these missing fields | Application gives an alert stating, “Please enter paying amount” |
|  |  | Currently paying amount is less than partial paying amount | Application must alert the user about the amount being less than the partial amount to be paid | Application gives an alert stating, “Paying amount should be at least partial due amount” |
|  |  | Invalid currently paying amount | Application must alert the user about the amount being invalid | Application gives an alert stating, “Paying Enter Valid paying amount” |
|  |  | Paying amount greater than total due amount | Application must alert the user about the amount being greater | Application gives an alert stating, “Paying amount should be less than total amount” |
|  |  | Negative Paying amount entered | Application must alert the user about the amount being negative | Application gives an alert stating, “Negative amount is not allowed!” |
|  | Remaining amount is updated in the database | Valid amount entered | Application must allow the user to update balance | Application shows remaining amount and gives a successful notification stating, “Payment Successful” and adds the transaction to payment history |

* Test Case  - UX18

Test Case Description : To verify that a back/home button is provided to the vendor (shopkeeper) to go back to dashboard

Pre-condition :User is a successfully logged-in vendor (shopkeeper) and clicks on Home button

Table 7.18 Back To Vendor Dashboard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User is on one of the tab screens | User clicks on the home button | Application must be redirect user to Login page | Application successfully redirects user to the dashboard |

* Test Case  - UX19

Test Case Description : To verify that a logout button is provided to the vendor (shopkeeper) to logout

Pre-condition :User is a successfully logged-in vendor (shopkeeper) and clicks on Logout button

Table 7.19 Vendor Logout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | When the user clicks on the logout button , the application will redirect the user to the Login page  Logging out removes all authentication information from the application | Object with current logged in user’s details | User must be redirected to Login page | Application successfully allows user to Logout of the application and redirects to the Login page |

* Test Case  - UX20

Test Case Description : To verify that a Registration page is provided for a consumer to register

Pre-condition : User selects consumer category and is a non-registered user

Table 7.20 Consumer Registration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | The first page rendered when the application starts is the Launching page containing two categories vendor (shopkeeper) and consumer. Selecting the consumer category renders the Login page.  Below there is a link to register for new users, User clicks New User ? Register  which opens the Registration page | Name, registered mobile number, Address, password not entered | Application must alert the user about these missing fields | Application alerts the user about these missing fields stating,   1. Missing name - “Please Enter Your Name” 2. Missing contact number - “Please Enter Your Contact Number” 3. Missing shop name - “Please Enter Your Shop Name” 4. Missing shop address - “Please Enter Your Address” 5. Missing password - “Please Enter Your Password” 6. Missing entry of password to verify it - “Please Re-Enter Your Password” |
|  |  | Invalid Contact Number | Application must alert the user with an error message | Application gives an alert stating, “Required 10 Digit Valid Contact Number” |
|  |  | Already  Registered Mobile  Number | Application must alert the user about the already  registered mobile number | Application gives an alert stating, “Contact Number Already Exists” |
|  | User’s contact number, password and details are successfully stored in the database | Valid fields  entered | Application must notify the user about successful registration | Application gives a notification stating, “Registration Successful” |

* Test Case  - UX21

Test Case Description : To verify that a Login page is provided for all consumers to login to Dashboard

Pre-condition : User selects consumer category and is a registered user

Table 7.21 Consumer Login

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | The first page rendered when the application starts is the Login page | RMN or/and Password not entered | Application must alert the user about these missing fields | Application alerts the user about these missing fields stating,   1. Missing RMN - “Please Enter Your Registered Mobile Number” 2. Missing password - “Please Enter Your Password” 3. Missing RMN and password - “Please Enter RMN And Password” |
|  |  | Unregistered contact number | Application must alert the user about contact number unregistered | Application gives an alert stating, “User Not Found! Please Register First To Login” |
|  |  | Invalid RMN/password | Application must alert the user with an error message | Application alerts the user about these invalid fields stating,   1. Invalid RMN - “Required 10 Digit Valid Contact Number” 2. Invalid Password - “Please Enter A Valid Password” |
|  | Consumer’s name and registered mobile number are successfully stored in the database | Valid Fields entered | Application must notify the user about successful login | Application gives a message stating, “Welcome” |

* Test Case  - UX22

Test Case Description : To verify that the Dashboard UI is provided to the consumer

Pre-condition : User is a successfully logged in consumer and is on the Dashboard which is also a default landing page after successful login

Table 7.22 Consumer Dashboard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | The first page rendered when the application starts after user logins the consumer category is the Dashboard | User is rendered on Dashboard | Application must display list of all vendor’s (shopkeeper’s) shop with their due amount to be paid, search bar to search a vendor and logout button | Application displays a list of all shop names of vendors with their RMN and total due amount. A search bar to search a vendor using their shop  name or contact, home button and logout button |
|  |  | User selects a vendor from the list | Application must  redirect to the My Udhaari page for that selected vendor’s shop | Application successfully renders to My Udhaari page for selected vendor |
|  |  | User clicks the search bar | Application must show a popup keypad on the screen to type the credentials of a vendor’s shop | Application shows keypad pop-up on the screen to type the vendor credentials and based on the search filters the vendor list |
|  | After clicking on any option on the Dashboard the user must be able to go to the directed page | User clicks the Logout button | Application must allow user to logout successfully and redirects to the Login page | Application successfully allows user to Logout of the application and redirects to the Login page |

* Test Case  - UX23

Test Case Description : To verify that a search bar is provided for a consumer to search a vendor (shopkeeper)

Pre-condition : User is a valid logged-in vendor (shopkeeper) and on a Dashboard of the app

Table 7.23 Consumer Dashboard – Search Bar

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | To select a specific vendor (shopkeeper)’s shop among all the vendors using a search bar | Invalid Shop name or contact | Application must show message for invalid input without any search results | Application gives a blank consumer list without any search results |
|  |  | Valid shop name | Application must display that vendor with their shop name | Application will display that vendors with shop name, RMN and total due amount highlighted in red (blocked account) and green (active account) |
|  | Consumer selects the vendor and successfully gets redirected to My Udhaari page | Valid vendor RMN | Application will display that vendor with their shop name | Application will display that vendor’s with shop name, RMN and total due amount highlighted in red (blocked account) and green (active account) |

* Test Case  - UX24

Test Case Description : To verify that UI  is provided to consumer for a selected vendor (shopkeeper) with options to navigate

Pre-condition : User is logged in consumer and have selected a vendor (shopkeeper)

Table 7.24 Consumer – Navigation Tab

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | The first page rendered when the user selects any vendor’s (shopkeeper’s) shop is the My Udhaari page of that vendor (shopkeeper) | User clicks my udhaari (Default Rendered page) | Application must  direct to the Udhaari Records page | Application successfully directs to the Udhaari Record’s page having all purchase and payment transactions of selected consumer |
|  |  | User clicks my account | Application must  direct to My Account page and should display all details of vendor as well as logged in consumer | Application successfully directs to the Account Details page containing selected consumer’s name, RMN, address, threshold, account status, threshold, account start date, billing start date, bill due date, last paid amount, total due amount, partial due amount |
|  | After clicking on any option on the screen the user must be able to go to the directed page | User clicks Logout | Application must direct to the Login | Application successfully allows user to Logout of the application and redirects to the Login page |

* Test Case  - UX25

Test Case Description : To verify that the records of purchase history is provided to the consumer of selected vendor (shopkeeper)

Pre-condition : User is a successfully logged-in consumer and is on My Udhaari page of a selected vendor (shopkeeper)

Table 7.25 My Udhaari – Purchase Transaction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User selects payment history with filter for year and month | User does not click purchase history | Application must by default show the latest records of all payment as well as purchase history | Application shows all the latest transactions of both payment as well as purchase history |
|  |  | User clicks purchase history | Application must show all the records of purchase transactions | Application shows all purchase transactions with latest transaction as the first one |
|  |  | User filters the year and month to see the purchase history accordingly | Application must show all the transactions based on the filtered year and month | Application shows all purchase transaction based on the filtered year and month |
|  |  | User clicks on one of the transactions | Application must show details about that transaction | Application shows detailed description of that transaction in a bill format |
|  |  | User filters an inactive month/year or invalid month/year  to see the purchase history | Application must notify the user about it | Application shows no records as the years were not active |

* Test Case  - UX26

Test Case Description : To verify that the records of payment history is provided to the consumer

Pre-condition : User is a successfully logged-in consumer and is on My Udhaari page of a selected vendor (shopkeeper)

Table 7.26 Payment Transaction

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User selects payment history with filter for year and month | User does not click payment history | Application must by default show the latest records of all payment as well as purchase history | Application shows all the latest transactions of both payment as well as purchase history |
|  |  | User clicks payment history | Application must show all the records of payment transaction | Application shows all payment transaction with latest transaction as the first one |
|  |  | User filters the active years and months to see the payment history | Application must show all the transactions based on the filtered year and month | Application shows all payment transaction based on the filtered year and month |
|  |  | User clicks on one of the transactions | Application must show details about that transaction | Application shows detailed description of that transaction in a bill format |
|  |  | User filters an inactive year or invalid year  to see the payment history | Application must notify the user about it | Application shows no records as the years were not active |

* Test Case  - UX27

Test Case Description : To verify that an option is provided to view account details of a consumer

Pre-condition : User is a successfully logged-in consumer and is on My Account page.

Table 7.27 My Account

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | This page is rendered when user clicks on  Account Details option | On rendering to Account Details page | Application should show all details of the consumer such as name, registered mobile number , date on which account is started, threshold, last paid amount, start date of account, due date ,total due amount and partial due amount | Application gives all details of consumer such as name, registered mobile number, date on which account is started, threshold ,last paid amount, start date of account, due date ,total due amount and partial due amount |
|  |  | User has paid all payments on time | Application should show active account sign | Application gives active account sign for that consumer |
|  | Retrieve all details of consumer from  database | User has not done payment | Application should show blocked account sign | Application gives active account sign for that consumer |

* Test Case  - UX28

Test Case Description : To verify that a back/home button is provided to the consumer to go back to dashboard

Pre-condition :User is a successfully logged-in consumer and clicks on Home button

Table 7.28 Back To Consumer Dashboard

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | User is on one of the tab screens | User clicks on the home button | Application must be redirect user to Login page | Application successfully redirects user to the dashboard |

* Test Case  - UX29

Test Case Description : To verify that a logout button is provided to the consumer to logout

Pre-condition : User is a successfully logged-in consumer and clicks on logout button

Table 7.29 Consumer Logout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Step No** | **Step Description** | **Test Data** | **Expected Result** | **Actual Result** |
| 1. | When the consumer clicks on the logout button , the application will redirect the consumer to the Login page.  Logging out removes all authentication information from the application | Object with current logged in user’s details | User is redirected to the Login page | Application successfully allows user to Logout of the application and redirects to the Login page |

Test Cases Execution

This part of the project describes, the process of executing the code and comparing the expected and actual results. The test cases executed are as follows,

* Test Case  - BackEnd01

Test Case Description : To verify that a registration page is provided to the vendor (shopkeeper) to register itself so that they can login

 Pre-condition :Server is running

Table 8.1 Backend Testing – Vendor Registration

|  |  |
| --- | --- |
| **Step Description** | Vendor (shopkeeper) provides Name, Contact Number, Shop Name, Shop Address, Set Password and clicks register button |
| **Test Data** | Name: Tabita  Contact Number: 9087689124  Shop Name: Cake Bakery  Shop Address: Sadashiv Peth, Pune  Password: tabita123 |
| **Expected Result** | The system allows the user to successfully register. |
| **Actual Result** | The system verifies the entered credentials with validation, if valid allows the user to register. |

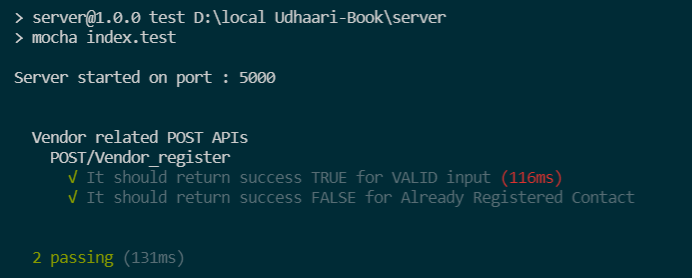


Figure 8.1 Backend Testing - Vendor Registeration

* Test Case  - BackEnd02

Test Case Description: To verify that a login page is provided to the vendor (shopkeeper)

Pre-condition: Server is running

Table 8.2 Vendor Login

|  |  |
| --- | --- |
| **Step Description** | Consumer provides Contact Number and  Password and clicks login button |
| **Test Data** | Registered Mobile Number : 9087689124  Password : tabita123 |
| **Expected Result** | The system allows the vendor to successfully login. |
| **Actual Result** | The system verifies the entered credentials with validation, if valid allows the vendor to access dashboard. |

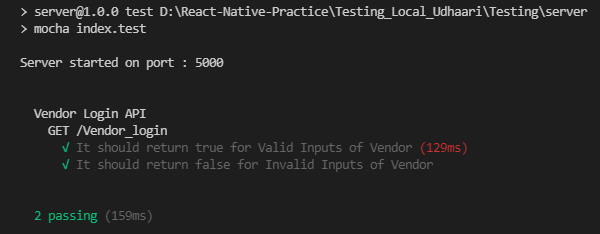


Figure 8.2 Vendor Login

* Test Case  - BackEnd03

Test Case Description : To verify that a dashboard page is provided to the vendor (shopkeeper) after successful login

Pre-condition : Server is running

Table 8.3 Vendor Dashboard

|  |  |
| --- | --- |
| **Step Description** | Vendor (shopkeeper) is at dashboard |
| **Test Data** | Vendor can see array of consumers with their name , contact and due amount |
| **Expected Result** | The system allows the vendor to search and display list of consumers |
| **Actual Result** | System by default display list of all consumers. Vendors can search for particular consumer and the system displays an array of consumers based on search. |

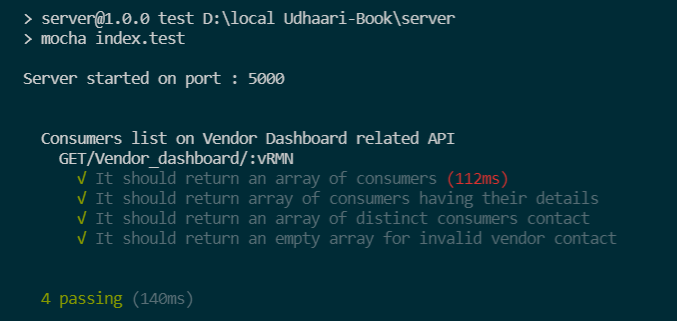


Figure 8.3 Vendor Dashboard

* Test Case  - BackEnd04

Test Case Description : To verify that a add consumer page is provided to the vendor (shopkeeper) to add new consumer

Pre-condition :Server is running

Table 8.4 Add New Consumer

|  |  |
| --- | --- |
| **Step Description** | Vendor (shopkeeper) clicks floating button and rendered at add consumer page then add details of consumer and click on add consumer button |
| **Test Data** | Threshold : 0.7  Start Date : 2021/05/12 |
| **Expected Result** | The System should add new consumer and rendered the vendor to dashboard |
| **Actual Result** | The system verifies the entered credentials with validation, if valid allows the vendor to add new consumer and rendered to dashboard |

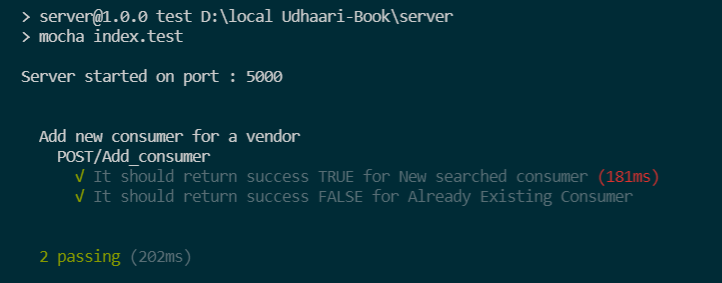


Figure 8.4 Add New Consumer

* Test Case  - BackEnd05

Test Case Description : To verify that a search bar is provided to the vendor (shopkeeper) to search for new consumer

Pre-condition : Server is running

Table 8.5 Vendor Dashboard – Search Bar

|  |  |
| --- | --- |
| **Step Description** | Vendor (shopkeeper) clicks on search bar and search consumer by their contact number |
| **Test Data** | Search a consumer in a search bar by consumers contact number |
| **Expected Result** | The System should display result according to availability of consumer |
| **Actual Result** | The System returns true if a new searched consumer is available and false if the searched consumer already exists. |

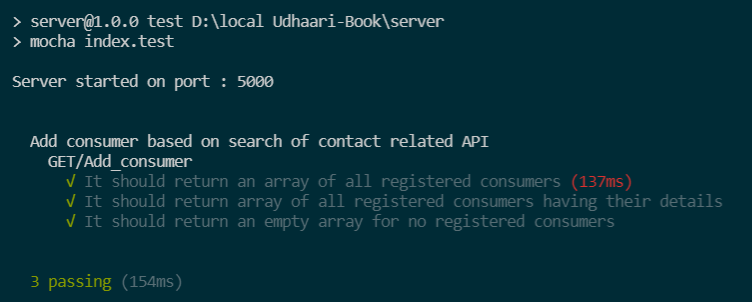


Figure 8.5 Search Bar For Vendor Dashboard

* Test Case  - BackEnd06

Test Case Description : To verify that a Account details page  is provided to the vendor (shopkeeper) on selecting consumer from dashboard

Pre-condition :Server is running

Table 8.6 Account Details

|  |  |
| --- | --- |
| **Step Description** | Vendor (shopkeeper) clicks on consumer  and rendered to account details page |
| **Test Data** | Display details of consumer such as name, address , contact number , status , threshold , account start date , billing start date , bill due date, last paid amount ,total due amount and partial due amount |
| **Expected Result** | The System should display all account details of consumer |
| **Actual Result** | The System returns all  details such as name , address , contact number , status , threshold , account start date , billing start date , bill due date, last paid amount ,total due amount and partial due amount for selected consumer |

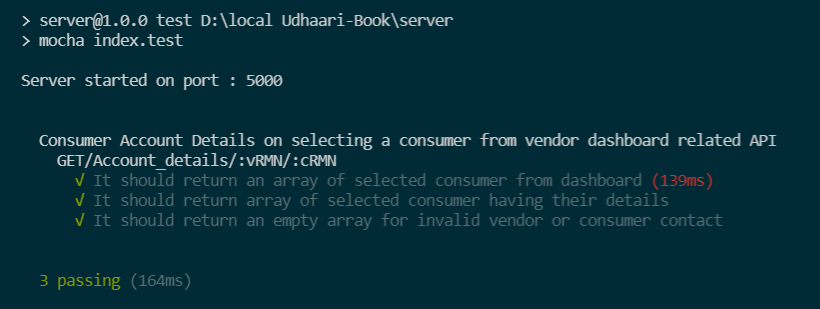


Figure 8.6 Account Details

* Test Case  - BackEnd07

Test Case Description : To verify that a vendor (shopkeeper) can edit name and threshold of particular consumer

Pre-condition :Server is running

Table 8.7 Edit Account

|  |  |
| --- | --- |
| **Step Description** | Vendor (shopkeeper) clicks on name or threshold fields and change details successfully |
| **Test Data** | Vendor clicks on name or threshold fields and then click on save changes button |
| **Expected Result** | The System should make changes and display success message |
| **Actual Result** | The system verifies the entered name or threshold with validation, if valid allows the vendor to make changes and display message “Name changed successfully” or “Threshold changed successfully” |

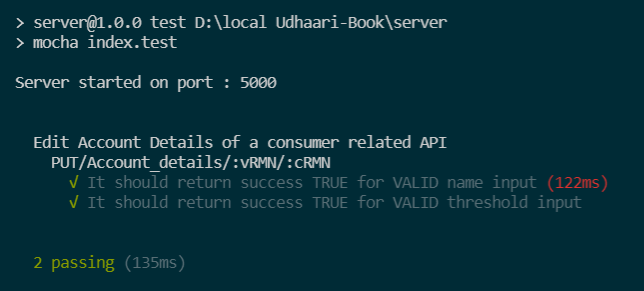


Figure 8.7 Edit Account

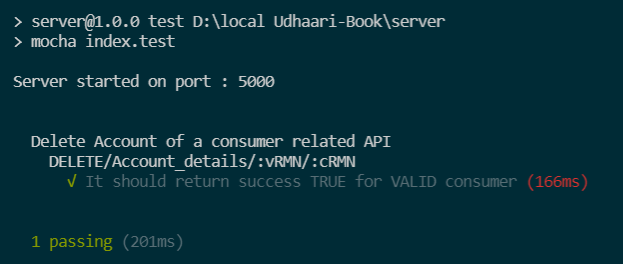
* Test Case  - BackEnd08

Test Case Description : To verify that a vendor (shopkeeper) can delete account  of particular consumer

Pre-condition :Server is running

Table 8.8 Delete Account

|  |  |
| --- | --- |
| **Step Description** | Vendor (shopkeeper) clicks on delete button and consumer get deleted |
| **Test Data** | Vendor clicks on delete button then system confirm to delete the account and after confirmation account will be deleted successfully |
| **Expected Result** | The System should allow vendor to delete selected account and after deletion redirect the vendor to dashboard |
| **Actual Result** | The system confirms vendor to delete particular account and if vendor make confirmation to delete account then system delete the account with message “Account deleted Successfully” and vendor redirects to the dashboard |

Figure 8.8 Delete Account

* Test Case  - BackEnd09

Test Case Description : To verify that a add products page is rendered for particular consumer

Pre-condition :Server is running

Table 8.9 Add Product

|  |  |
| --- | --- |
| **Step Description** | Consumer purchase products and vendor (shopkeeper) add products details to bill list |
| **Test Data** | product\_id : 8  quantity : 2  date\_purchase : ‘2021/05/12’  time\_purchase : ‘15:30:38’  total\_price : 1200  tr\_id : 2342156781  total\_amount : 2200 |
| **Expected Result** | The System should allow vendor to add different products with their details to bill list |
| **Actual Result** | The system allows vendor to add product name , price , quantity and total amount to bill list |

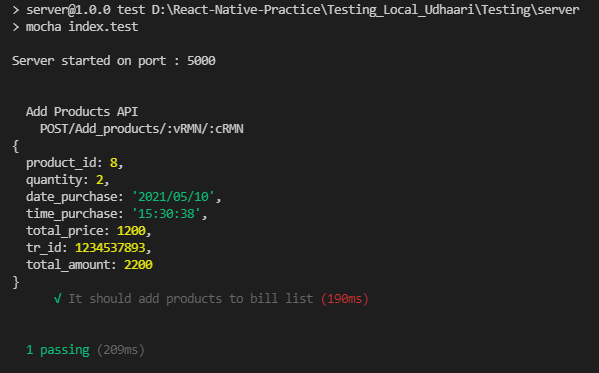


Figure 8.9 Add Product

* Test Case  - BackEnd10

Test Case Description : To verify that a total amount is display for particular consumer

Pre-condition :Server is running

Table 8.10 Total Amount/Current Bill

|  |  |
| --- | --- |
| **Step Description** | Total Due amount of particular consumer should be displayed |
| **Test Data** | Total due amount of consumer |
| **Expected Result** | The System should display total due amount of selected consumer |
| **Actual Result** | The system displays total due amount of selected consumer and value of total due amount changes dynamically when consumer purchase any new products |

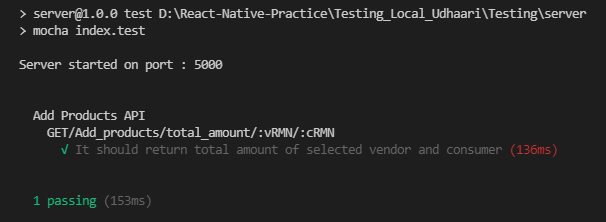


Figure 8.10 Total Amount/Current Bill

* Test Case  - BackEnd11

Test Case Description : To verify that a products list page is rendered when vendor (shopkeeper) clicks on view all products

Pre-condition :Server is running

Table 8.11 View Products

|  |  |
| --- | --- |
| **Step Description** | List of products with their name and base price is displayed |
| **Test Data** | Search bar to search for particular product, floating button to open modal add new products |
| **Expected Result** | The System should display list of products with their name and base price |
| **Actual Result** | The system displays search bar to search for particular product, floating button to open modal add new products and list of products with their name and base price |

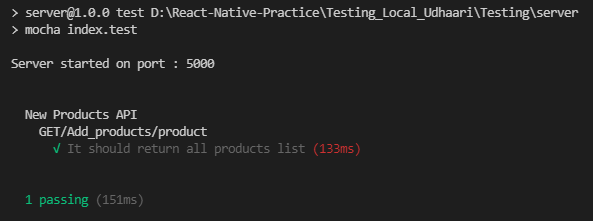


Figure 8.11 View Products

* Test Case  - BackEnd12

Test Case Description : To verify that a new product is added by vendor (shopkeeper) to product list

Pre-condition :Server is running

Table 8.12 Add A New Product

|  |  |
| --- | --- |
| **Step Description** | New Products is added to product list and if product is already exists then give a error message |
| **Test Data** | product\_name : ‘maida’  base\_price : 83 |
| **Expected Result** | The system should return true for adding new product to product list and false for already existing the product with same base price |
| **Actual Result** | The system return message “Product added successfully” for adding new product to product list and “Product already exists” for already existing the product with same base price |

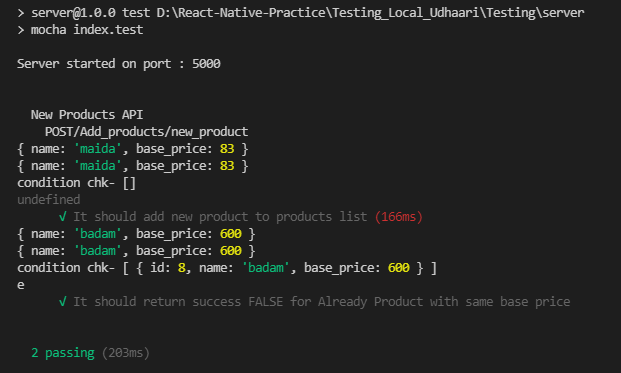


Figure 8.12 Add A New Product

* Test Case  - BackEnd13

Test Case Description : To verify that a vendor (shopkeeper) is rendered to udhaari records page

Pre-condition : Server is running

Table 8.13 Udhaari Records

|  |  |
| --- | --- |
| **Step Description** | It should display 3 options such as all , purchase and payment |
| **Test Data** | Date picker to select particular month and year |
| **Expected Result** | The system should return all transaction history of current month by default along with total due amount of that consumer |
| **Actual Result** | The system displays  all transaction history of current month by default along with total due amount of that consumer |

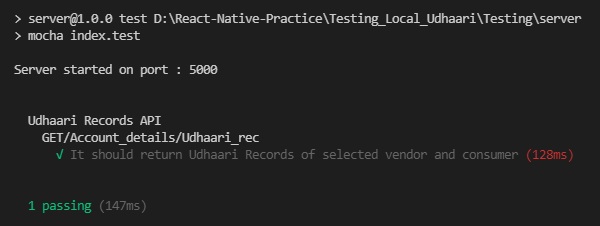


Figure 8.13 Udhaari Records

* Test Case  - BackEnd14

Test Case Description : To verify that vendor (shopkeeper) can view all transaction history of selected consumer

Pre-condition :Server is running

Table 8.14 Transaction History

|  |  |
| --- | --- |
| **Step Description** | It should display all transaction history of selected consumer |
| **Test Data** | All transaction history along with purchase and payment card of selected consumer |
| **Expected Result** | The system should return all transaction history along with purchase and payment card of selected consumer |
| **Actual Result** | The system displays all transaction history by default along with purchase and payment card of selected consumer |

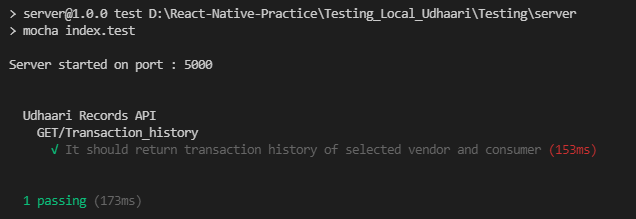


Figure 8.14 Transaction History

* Test Case  - BackEnd15

Test Case Description : To verify that vendor (shopkeeper) can view purchase bill of selected consumer

Pre-condition :Server is running

Table 8.15 Purchase Transaction

|  |  |
| --- | --- |
| **Step Description** | It should display purchase bill of selected consumer |
| **Test Data** | View all details and close button |
| **Expected Result** | The system should return all purchase details of selected transaction id of consumer |
| **Actual Result** | The system displays purchase bill along with transaction id, date, product, price, quantity and total price |

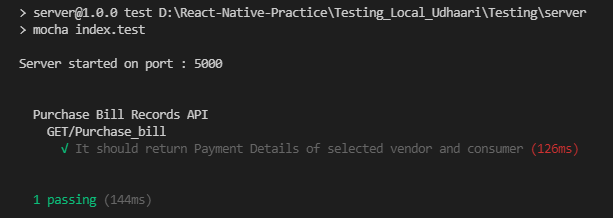


Figure 8.15 Purchase Transaction

* Test Case  - BackEnd16

Test Case Description : To verify that vendor can view all  purchase bill of selected consumer

Pre-condition :Server is running

Table 8.16 Purchase History

|  |  |
| --- | --- |
| **Step Description** | It should display all purchase bill of selected consumer |
| **Test Data** | View all purchase bill in card format |
| **Expected Result** | The system should all purchase bill of selected consumer ordered by last bill will display first |
| **Actual Result** | The system displays all purchase bill along with transaction id , date, product , price , quantity and total price of selected consumer ordered by last bill will display first |

### 

Figure 8.16 Purchase History

* Test Case  - BackEnd17

Test Case Description : To verify that vendor (shopkeeper) can view payment bill of selected consumer

Pre-condition :Server is running

Table 8.17 Payment Transaction

|  |  |
| --- | --- |
| **Step Description** | It should display payment bill of selected consumer |
| **Test Data** | View all details and close button |
| **Expected Result** | The system should return all payment details of selected transaction id of consumer |
| **Actual Result** | The system displays purchase bill along with transaction id , date, total amount , paid amount and carry forward amount |

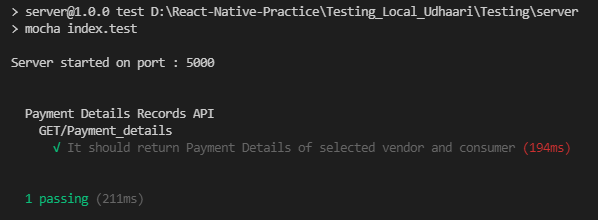


Figure 8.17 Payment Transaction

* Test Case  - BackEnd18

Test Case Description : To verify that vendor can view all  payment bill of selected consumer

Pre-condition :Server is running

Table 8.18 Payment History

|  |  |
| --- | --- |
| **Step Description** | It should display all payment bill  of selected consumer |
| **Test Data** | View all details and close button |
| **Expected Result** | The system should return all payment details of selected transaction id of consumer ordered by last bill will display first |
| **Actual Result** | The system displays purchase bill along with transaction id, date, total amount, paid amount and carry forward amount of selected consumer ordered by last bill will display first |

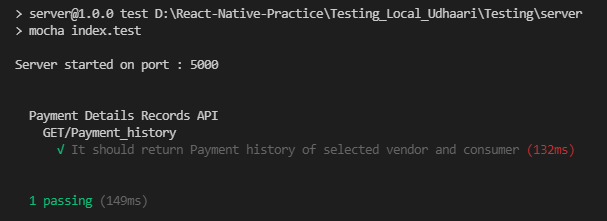


Figure 8.18 Payment History

* Test Case  - BackEnd19

Test Case Description : To verify that vendor (shopkeeper) can get threshold of selected consumer

Pre-condition :Server is running

Table 8.19 View Threshold

|  |  |
| --- | --- |
| **Step Description** | It should display threshold of selected consumer |
| **Test Data** | Vendor (shopkeeper) can get threshold of selected consumer |
| **Expected Result** | The system should return threshold of selected consumer |
| **Actual Result** | The system displays threshold of selected consumer |

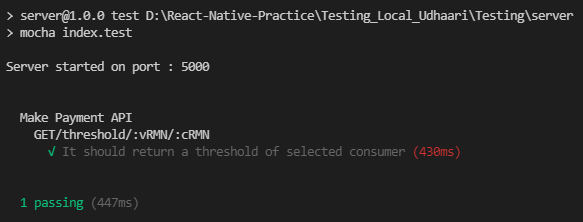


Figure 8.19 View Threshold

* Test Case  - BackEnd20

Test Case Description : To verify that vendor (shopkeeper) can view make payment page of selected consumer

Pre-condition :Server is running

Table 8.20 Make Payment

|  |  |
| --- | --- |
| **Step Description** | Allows vendor (shopkeeper) to make payment of selected consumer |
| **Test Data** | date: ‘2021/05/12’  paying\_amount: 1200 |
| **Expected Result** | The system should display all details of consumer and return true for valid input transactions |
| **Actual Result** | The system displays total due amount, partial due amount, paying date, paying amount and remaining amount of selected consumer and make payment button |

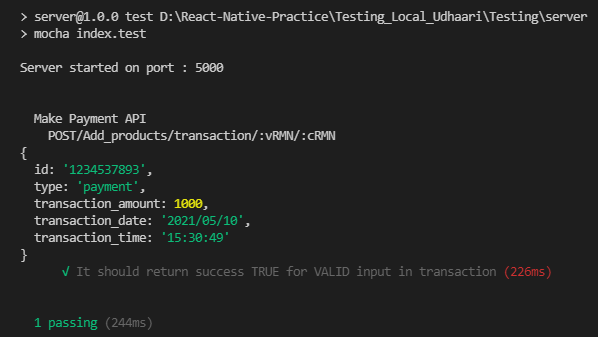


Figure 8.20 Make Payment

* Test Case  - BackEnd21

Test Case Description : To verify that vendor (shopkeeper) can change payment details of selected consumer

Pre-condition : Server is running

Table 8.21 Change Payment Details

|  |  |
| --- | --- |
| **Step Description** | Allows vendor (shopkeeper) to change payment details of selected consumer |
| **Test Data** | consumer\_contact: 9881779683  vendor (shopkeeper)\_contact: 9880765432  total\_amount: 2500  paid\_amount: 1500  remaining\_amount: 1000  transaction\_date: ’2021/05/12’  transaction\_time: ’15:30:49’  tr\_id: 1289576032 |
| **Expected Result** | The system should display all details of consumer and return true for valid input change in data |
| **Actual Result** | The system posts the consumer contact, vendor contact, total amount, paid amount, remaining amount , transaction date , transaction time and tr\_id into database |

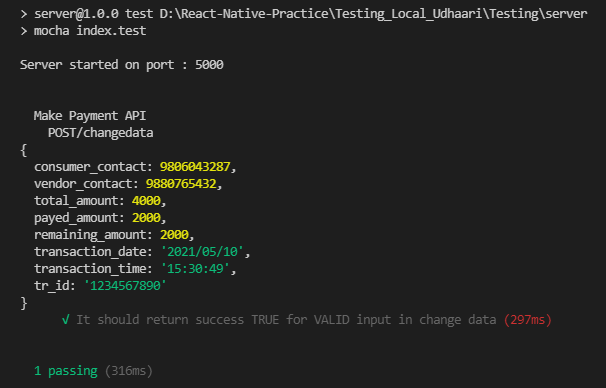


Figure 8.21 Change Payment Details

* Test Case  - BackEnd22

Test Case Description : To verify that vendor (shopkeeper) can update payment details of selected consumer

Pre-condition :Server is running

Table 8.22 Update Payment Details

|  |  |
| --- | --- |
| **Step Description** | Allows vendor (shopkeeper) to update payment details of selected consumer |
| **Test Data** | Update payment details in database |
| **Expected Result** | The system should update all details of consumer and return true for valid inputs |
| **Actual Result** | The system updates all details of consumer in database and return  true for valid inputs |

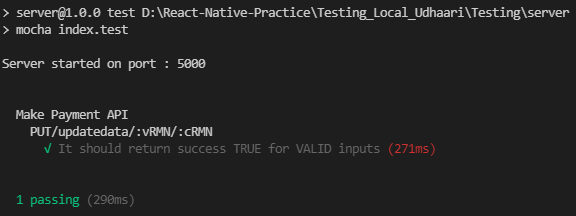


Figure 8.22 Update Payment Details

* Test Case  - BackEnd23

Test Case Description : To verify that a registration page is provided to the consumer to register itself so that they can login

Pre-condition :Server is running

Table 8.23 Consumer Registration

|  |  |
| --- | --- |
| **Step Description** | Consumer  provides Name, Contact Number, Address, Set Password and clicks register button |
| **Test Data** | Name :Dipika  Contact Number : 8894562213  Address : Panchavati Karanja , Nashik  Password : dipika123 |
| **Expected Result** | The system allows the user to successfully register |
| **Actual Result** | The system verifies the entered credentials with validation, if valid allows the user to register. |

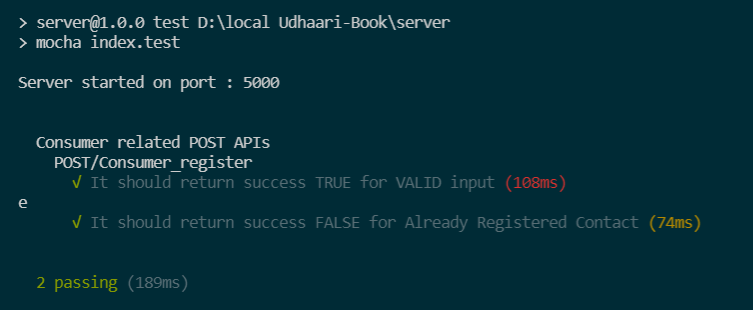


Figure 8.23 Consumer Registration

* Test Case  - BackEnd24

Test Case Description : To verify that a login page is provided to the consumer to login.  Pre-condition :Server is running

Pre-condition :Server is running

Table 8.24 Consumer Login

|  |  |
| --- | --- |
| **Step Description** | consumer provides Contact Number and  Password and clicks login button |
| **Test Data** | Contact Number : 8894562213  Password : dipika123 |
| **Expected Result** | The system allows the consumer  to successfully login. |
| **Actual Result** | The system verifies the entered credentials with validation, if valid allows the consumer  to access the dashboard. |

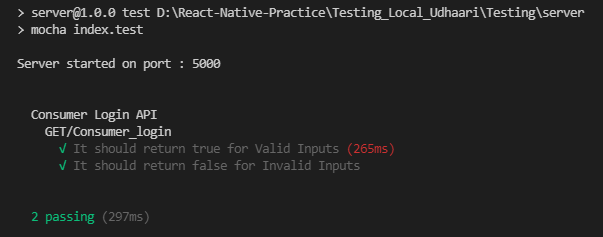


Figure 8.24 Consumer Login

* Test Case  - BackEnd25

Test Case Description : To verify that a dashboard page is provided to the consumer after successful login

Pre-condition :Server is running

Table 8.25 Consumer Dashboard

|  |  |
| --- | --- |
| **Step Description** | consumer is at dashboard page |
| **Test Data** | consumer can see array of vendors with their shop name , contact and due amount |
| **Expected Result** | The system allows the consumer to search and display list of vendor (shopkeeper) |
| **Actual Result** | System by default display list of all vendors. Consumer can search for particular vendor (shopkeeper) and the system displays an array of vendors based on search. |

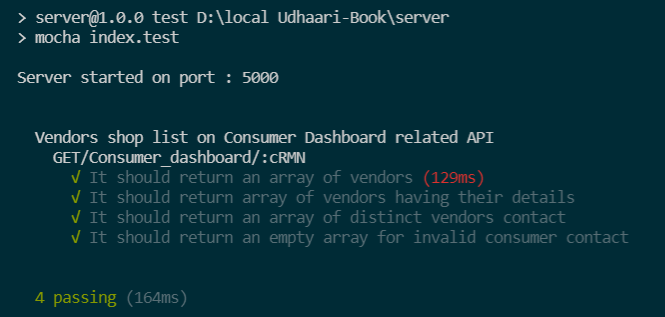


Figure 8.25 Consumer Dashboard

* Test Case  - BackEnd26

Test Case Description : To verify that a My Account  page  is provided to the consumer on selecting vendor (shopkeeper)  from dashboard

Pre-condition :Server is running

Table 8.26 My Account

|  |  |
| --- | --- |
| **Step Description** | Consumer clicks on vendor (shopkeeper) and rendered to My Account page |
| **Test Data** | Display details of consumer such as name, address, contact number, status, threshold, account start date, billing start date, bill due date, last paid amount, total due amount and partial due amount |
| **Expected Result** | The System should display all account details of consumer |
| **Actual Result** | The System returns all details such as name, address, contact number, status, threshold, account start date, billing start date, bill due date, last paid amount, total due amount and partial due amount for selected consumer |



Figure 8.26 My Account

Defect Tracking and analysis

It includes all the defects found during testing. Description of the defect., analysis of what the defect was, which module/area the defect was found in, what impact it had, what’s the finding from investigation, what was the cause of the defect and the resolution.

Table 9 Defects, Cause & Resolution

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.**  **No.** | **Defects** | **Cause** | **Resolution** |
|  | Module – Account\_details, Udhaari\_records & My\_udhaari  Date to be entered in the database unmatched during deployment | Local database time zone set differed | Changes done in the online database by setting local timezone |
|  | Module – Udhaari\_records  Recent transaction sorted based on transaction ID | When sorted using date gave unexpected result | Transaction ID generated based on date/time hence sorts recent transaction ID in descending order |
|  | Modules – All modules that include values to be taken as input  State variables not cleared after use | The state variables once submitted does not get cleared automatically | As the task gets completed the state variables are re-initialized with their initial states |
|  | Module – Vendor\_register, Consumer\_register, Add\_consumer  At a time only one user allowed to register/added | Navigate to previous screen enabled after submission hence screen changes after submission | Introduced back button wherever required so user can register/add as many users required at an instance |
|  | Module - Add\_products  Once vendor add product it cannot be deleted | No option to delete purchased product once added in the add products table | Option provided to delete particular product on each row |

Deployment & Implementation

This part refers to the process of making the application work on a target device, whether it be a test server, production environment or a user's computer or mobile device.

The project is setup and deployed as follows,

* Setup all the files in the project directory for the heroku build structure
* Configure main Server file
* Configure Database Connection
* Configure production and local connectivity using process.env
* Setup Scripts in package.json
* Set heroku start and post-build
* Set up engines in package.json and add catchall method
* Install Heroku CLI
* Deploy to Heroku
* Login to heroku
* Create new project on heroku
* Add add-on : Heroku Postgres (Hobby Dev - Free version)

Configure database - create tables

* Go to deploy section in created section on heroku

Connect to Git account

Select Your Project repository

Git push heroku master

Project is deployed

Limitations & Future Enhancements

Limitations

It gives information about what the defined system is is unable to do as of its present state which occur through variations in collected data.

* Payment mode is offline that needs to be handled by vendor (shopkeeper)
* This project is developed for mobile application, hence it is not completely responsive to web application
* User has to view app in only english language
* User does not get any notification when vendor add that consumer
* Vendor wise product list is not available

Future Enhancement

It suggests almost all the essential requirements covered further requirements and improvements can easily be done since the coding is mainly structured or modular in nature. Here is a list of few enhancements that maybe implemented in the future,

* User can select language preferences
* Vendor (shopkeeper) can add product name by scanning QR code of product
* Consumer can make online payment to vendor
* Consumer can get notification if vendor added that consumer to their consumer list
* Consumer can add different products to Todo list and send list to particular vendor
* If consumer’s due date is expired then system should automatically block that consumer’s account and prevent vendor to add purchase products to that list until previous payment done by that consumer
* Vendor can publish advertisements and different offers to the consumer

Conclusion

This internship was a professional experience in the world of computer science. In a nutshell, it was an excellent and rewarding experience. All weekly as well as monthly tasks offered opportunities to learn and develop knowledge and skills in many areas and also helped in gaining a lot of experience in web, database and cross platform development notably by learning a new and innovative knowledge in react native app development.

Finally, this internship allowed autonomously, to acquire skills in design and architecture application which helped in exploring more in the technology world.

References

* <https://docs.expo.io/>
* <https://reactnavigation.org/>
* <https://devcenter.heroku.com/categories/deployment>
* <https://docs.expo.io/versions/latest/sdk/securestore/>
* <https://templates.office.com/en-in/simple-gantt-chart-tm16400962>
* <https://reactnative.dev/docs/modal>
* <https://www.npmjs.com/package/react-native-modal-datetime-picker>
* <https://openbase.com/js/react-native-input-spinner>
* <https://mochajs.org/>
* <https://www.chaijs.com/plugins/chai-http/>
* <https://docs.swmansion.com/react-native-gesture-handler/docs/>
* Roger Pressman's Software Engineering: A Practitioner's Approach Book

Glossary:

This is an alphabetical list of words relating to a specific subject, text, or dialect, with explanations; a brief dictionary.

Table 10 Glossary

|  |  |
| --- | --- |
| **Term** | **Meaning** |
| API | Stands for *Application Programming Interface*. It is a set of functions and procedures that allow for the creation of applications that access data and features of other applications, services, or operating systems. |
| Assumption | Any project factor that is considered to be true, real, or certain without empirical proof or demonstration. |
| Bug | A failure or flaw in software program. |
| Caveat | A warning or proviso of specific stipulations, conditions, or limitations. |
| CLI | Stands for *Command Line Input.* |
| Development lifecycle | A framework defining tasks performed at each step in the software development process. |
| Effort estimation report | A comparison of estimated time for tasks vs actual time taken to complete them. |
| Entity | Something that has a separate and distinct existence and objective or conceptual reality. |
| Feasibility study | Assessment of practicality of proposed project or plan. |
| Feed | Content on a social media site. |
| Firebase | A platform developed by Google for creating mobile and web applications. |
| Front end development | The practice of converting data to a graphical interface, through the use of HTML, CSS, and JavaScript. |
| Gantt chart | A graphical description of a project schedule. |
| Git | Software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source |
| GitHub | A provider of Internet hosting for software development and version control using Git. |
| GUI | *Stands for Graphical User Interface. It* allows users to interact with different electronic devices using icons and other visual indicators. |
| Homepage | Application page that Users are redirected to, after successful login. |
| JavaScript | A lightweight, interpreted programming language designed for creating network-centric applications. |
| Lazy loading | A technique that defers loading of non-critical resources at page load time. |
| Libraries | A collection of non-volatile resources used by computer programs, often for *software* development. |
| Location picker | Component that allows Users to select a location. |
| Market research | The process of gathering information about current market, target audience and their needs. |
| Middleware | A software that bridges gaps between other applications, tools, and databases. |
| MoSCoW | Stands for Must, Should, Could & Would-have. It is a prioritization technique used to reach a common understanding of the product. |
| NoSQL | Non-tabular database. |
| Operational feasibility | the ability to utilize, support and perform the necessary tasks of a system or program. |
| Persona | An example of the kind of person who would interact with a system. |
| Post | An entry in the Feed, where User can add information about procured items. May or may not contain images. |
| Power-Interest grid | A matrix used for categorizing stakeholders during a change project to allow them to be effectively managed. |
| Primary key | A unique identifier for each record. |
| Product backlog | A prioritized features list, containing short descriptions of all functionality desired in the product. |
| Prototyping | An experimental process where *design* teams implement ideas into tangible forms from paper to digital. |
| RACI matrix | Stands for *Responsible, Accountable, Consulted & Informed.* It is a chart used to map each team member’s roles and responsibilities. |
| React JS | *JavaScript* library for building user interfaces. |
| Redux | State container for JavaScript applications. |
| Request | An entry in the Feed, where a User asks for a certain item from a certain location. May or may not contain images. |
| Requirement | Description of what a system should do. |
| Scrum cycle | Short development cycle in Agile methodology |
| SEO | Stands for *Search Engine Optimization*. It is the process of optimizing your online content so that a search engine likes to show it as a top result for searches of a certain keyword. |
| Stakeholder | A party that can either affect or be affected by a business. |
| Stakeholder engagement | Process by which companies communicate and get to know their stakeholders. |
| Stand-up meeting | Short, time-boxed team status check, typically at the start of the day. |
| System boundaries | Conceptual boundary that logically groups system components |
| Technical feasibility | The process of proving that the concept is technically possible. |
| Test case | A set of actions executed to verify a particular feature or functionality of an application. |
| Time feasibility | A measure of how reasonable the project timetable is. |
| Udhaari | Product/money taken on credit. |
| Unit testing | A type of software testing where individual units or components of a software are tested. |
| User story | A description of a software feature from an end-user perspective. |
| Value creation | Creating value benefits for stakeholders. |
| Version control | The practice of tracking and managing changes to software code. |
| Wrap-up meeting | Short, time-boxed team status check, typically at the end of the day. |