

RENTAL INVENTORY MANAGEMENT SYSTEM

Major Project Report

Submitted for Assessment for 6th semester.

PRESENTED BY

AVILASH BANERJEE

Registration number: 151170510009 of 2015-16

Roll Number: 11701015009

MEGHNAD SAMADDAR

Registration number: 151170510028 of 2015-16

Roll Number: 11701015028

RITARAJ PAL

Registration number: 151170510035 of 2015-16

Roll Number: 11701015035

Under the supervision of

Manas Ghosh

Asst. Professor at RCC Institute of Information Technology



At RCC Institute of Information Technology affiliated to Maulana Abul Kalam Azad University of
Technology, Canal South Road, Beliaghata, Kolkata-700015

May 2018



RENTAL INVENTORY MANAGEMENT SYSTEM SOFTWARE DESIGN DOCUMENT

By

Avilash Banerjee
Meghnad Samaddar
Ritaraj Pal



RCC INSTITUTE OF INFORMATION TECHNOLOGY

KOLKATA-700015, INDIA

**CERTIFICATE**

This is to certify that the project titled Rental Inventory Management System submitted by Avilash Banerjee (Roll number 11701015009 of MCA Department), Meghnad Samaddar (Roll number 11701015028 of MCA Department), Ritaraj Pal (Roll number 11701015035 of MCA Department) has been prepared under my/our supervision for the major project assessment, 6th semester.

[Name of the Guide]

Manas Ghosh

Asst. Professor, Dept. of CA, RCCIIT, Kolkata

Countersigned by,

[Name of the Head of Department]

Arup Kumar Bhattacharjee

Dept. of CA, RCCIIT, Kolkata

Declaration by Author(s)

This is to declare that this report has been written by me/us. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be plagiarized, I/we are shall take full responsibility for it.

TEAM MEMBERS

Avilash Banerjee

Ritaraj Pal

Meghnad Samaddar

ACKNOWLEDGEMENT

We express our sincere gratitude to Professor Manas Ghosh (Asst. Professor of Department of CA, RCCIIT) for extending his valuable time to guide us to take up this internship (and the associated project) and see it through.

Signature of the student,

Avilash Banerjee

Meghnad Samaddar

Ritaraj Pal

RCC INSTITUTE OF INFORMATION TECHNOLOGY

KOLKATA-700015, INDIA

**CERTIFICATE OF ACCEPTANCE**

This is to certify that the project titled Rental Inventory Management System submitted by Avilash Banerjee (Roll number 11701015009 of MCA Department), Meghnad Samaddar (Roll number 11701015028 of MCA Department), Ritaraj Pal (Roll number 11701015035 of MCA Department) is hereby recommended to be accepted for assessment for 6th Semester Examination in MAKAUT.

Name of the Examiner(s): Signature with Data

Date: _____

Table of contents

1.	Abstract	7
2.	Introduction	8
2.1.	Purpose	8
2.2.	Scope	8
2.3.	Definitions, Acronyms and Abbreviations	8
2.3.1.	Design Methodologies	8
2.3.2.	Software Technologies	9
2.3.3.	Other entities	10
3.	Feasibility Study	11
4.	Design Overview	13
4.1.	Description of Problem	13
4.2.	Technologies Used	14
4.3.	Constraints of Use	14
4.4.	System Architecture	15
4.4.1.	Database Architecture	16
4.4.2.	Server Architecture	23
4.5.	System Operations	31
5.	Requirements Traceability	36
6.	User Interface	39
7.	Data Model and Storage	48
7.1.	UML diagrams	48
7.1.1.	Use Case Diagram	48
7.1.2.	Activity Diagram	49
7.2.	Functional Diagrams and Algorithms	50
7.2.1.	React Component tree	50
7.2.2.	Data Flow Diagram	51
7.2.3.	E-R Diagram	55
7.2.4.	Flowcharts and Algorithms	56
8.	Advantages and Disadvantages of Proposed System.	68
9.	Future Scope	70
10.	Conclusion	71
11.	Bibliography	72

1: Abstract

This project report sufficiently describes the proposed 'Rental Inventory Management System' to determine its feasibility, usability and modular functionality. The client company 'Computer Exchange' experienced difficulties with their existing system wherein they used a windows application to keep track of their rental inventory, excel sheets to prepare reports and manual preparation of rental challans for their customers. This proposed system seeks to combine all these into a dynamic framework where the users would be able to create a repository of their customers, keep track, manage and edit their inventory products, create challans and invoices and prepare day to day reports for intra-office purposes; all within the same system. It uses a client-server model with a connected database to keep track of the various inventories and customers amongst the various branches of the office. The front end is modeled as a website so as to ensure user-friendly interfacing, even for their non technical personnel. The system has been made as a single version (Version 1) with space for future expansions of the system to include more high-end functionality like notification managers and payment portals. The feasibility studies, design, UML diagrams, advantages and disadvantages have been properly described in this document.

2: Introduction

2.1: Purpose

The purpose of this document is to describe the design and implementation of the Rental Inventory Management System website for the computer rental company 'Computer Exchange' as specified in the Software Requirement Specification document provided. The 'Inventory Management System' is designed to manage the day to day business activities of the said organization.

2.2: Scope

This document describes the implementation details of the Rental Inventory Management System, which includes the feasibility study, summary of functional and non-functional requirements as specified in the SRS, details of design and algorithms used in the development process and the application areas and use cases of the software.

This document however, will not include any source code, test cases or detail of testing procedures. For the details of using the software and utilizing its features, the reader is requested to consult the User Manual.

2.3: Definitions, Acronyms and Abbreviations

2.3.1: Design Methodologies

SRS Software Requirement Specifications. SRS stands for Software Requirement Specifications. It is responsible for defining in an unambiguous and concise way the functional and non-functional requirements of the software under development. It fully explains what the software is expected to do and serves as a binding contract between the developer and the client.

SDD software design document or (SDD) is a document responsible for providing the blueprint and the structure of the software to be developed in detail. It is used to explain the design details of the project to the software development team.

ACID It is an acronym for Atomicity, Consistency, Isolation and Durability. These are the desired properties for a database management system transaction. Atomicity dictates that the transactions should be performed completely or not at all. Consistency requires the system to stay in a stable and consistent state before or after a transaction. Isolation suggests that each

transaction must not affect or be affected by any other transaction. Durability ensures that the database should be recoverable, even after a failure condition occurs.

API Stands for Application Program Interface. It is a set of tools and methods that are utilised by the developers to develop the system. It works as an interface between different components of the software.

2.3.2: Software Technologies

React.js React.js or ReactJS is a javascript library which is used to dynamically develop frontend applications for web pages. It is used in the development of web based applications or mobile applications.

Node.js It is a javascript based platform that is used to create complex network applications. Node.js is an asynchronous, efficient library to run code server-side before sending them to the web application for rendering. Node.js has been used in this project for writing APIs that access/modify the database.

MySQL It is an efficient, easy to use database management system which uses SQL.

Heroku It is a popular cloud-based web hosting service which is used to deploy our server for data transfer.

Amazon Web Services (AWS) It is a state-of-the-art web hosting service that is perfect for commercial use and is not expensive for hosting web applications.

Bitbucket It is basically the git used by teams which provides functions and features for collaborative development efforts and version management.

2.3.3: Other entities

Computer Exchange Computer Exchange is a Kolkata based company that started in 1986 and are involved in the rent and hire business. It is one of the most popular locations for the rental and purchasing of IT equipment. They maintain goodwill amongst their old customers which in turn have increased their business several folds. The rental division fulfills the need of multiple IT companies across India.

Rental Inventory Management System Despite having a generic name, Rental Inventory management system is the name of the web application this document is detailing. The said web application is made by Dirac Business Solutions for the client ‘Computer Exchange’.

User The users of the web application, that is, the ‘Computer Exchange’ office staffs. The employees will be able to maintain a record of their products in the inventory, create and manage rental records, prepare challans, review records and create new repositories whenever required.

3: Feasibility Study

The following is a brief summary of the Feasibility Study Report for the Rental Inventory Management System for documentation purposes.

Technical Feasibility

Upon analysing the technical feasibility of this project, we concluded that we require the state of the art softwares and tools that are available today. We decided upon using React js framework as front end to provide fast and seamless user interface and navigation. Node js as back end server to process all the API calls and interactions with the database, with fast and asynchronous service. MySql was decided to be used as database as it is well known and easy to work with. All these tools were chosen to enable fast prototyping and deployment.

In the initial development process, Heroku was chosen as a platform for hosting the web application, which will be changed to Amazon Web Services before deployment. And Bitbucket was used for version control and management.

Therefore it was concluded that the project was technically feasible.

Economical Feasibility

The development cost of this application was decided to be kept as low as possible. Therefore almost all the softwares and tools that are used were completely free. Linux Mint OS as operating system, React js, Node js, MySql, Heroku services and Bitbucket was utilized as a result. It was decided to use Amazon Web Services upon deployment which is a paid service.

Therefore it was concluded that the project was economically feasible.

Legal Feasibility

Upon analysis it was concluded that there was no current legislation or prior commitments of the organization that will affect the project. Therefore It was legally feasible to pursue the project.

Operational Feasibility

This project is specifically designed for the use of the ‘Computer Exchange’ office staff and other intra-office purposes. The employees will be able to maintain a record of their products in the inventory, create and manage rental records, prepare challans, review records and create new repositories whenever required. The product has been made as an user-friendly software so that even the non technical staff can use it without much difficulty.

The current system in place is a platform dependent application software that relies heavily on manual labour and management utilizing physical documents as a means of communicating with different subsystems. The new software will therefore be a much needed improvement and therefore it is operationally feasible.

Schedule Feasibility

Upon analyzing the functional and non-functional requirements and the number of available human resource it was concluded that by using Agile development methodology we can deliver the version one of the product within 3 months. The team included one Project Manager, one Database Administrator, one lead developer and three interns. Later due to employee turnover the development team was reduced to one Project Manager and two interns which caused revision of the schedule and delay on deployment of version one.

4: Design Overview

4.1: Description of Problem

The purpose of this system is to create a repository of assets and clients of Computer Exchange and make the job of managing inventory and customers easier. The Inventory Management System that is used currently by Computer Exchange is an outdated desktop based legacy software that requires a lot of manual labour and paper trail for inventory management and record keeping. The reports that are required by the upper management are generated manually by the help of excel sheets, which are printed out and kept as physical record. This process is tedious and requires a lot of time and effort to generate sub optimal reports that are not easy to read.

The Rental Inventory Management software that we are developing aims to overhaul the whole system and utilize the internet and the portability of web based application to create an unified system that can be used by all the employees of Computer Exchange. The system aims to eliminate the manual labour and record keeping as much as possible, and furthermore it can be run on any device that can run a JavaScript supported browser, which can be a computers, phones, tablets. The interface should be easy to use and navigate, while providing useful informations and reports in a compact manner.

One of the most important feature required by the client was the ability to assemble or disassemble assets that are rented. Suppose Desktop is an asset, then the client wanted the feature that enables them to remove its parts like RAM or Motherboard and Replace them by new ones as per request of the buyers or renters. Therefore the removed component will be added to inventory as new asset and the part that it gets replaced with is taken from the existing inventory and added to the Desktop. These changes will be reflected in the challan and as well as on the record.

Another challenge in the development of this system was to create a generalized way of introducing new assets. Computer Exchange deals with renting and acquisition of various kinds of assets, that is hardwares and softwares. Therefore it was necessary to create an interface where new types of assets can be defined by the employees by using the system itself, so that the system can adapt to new types of assets as new technologies are developed or new types of assets are acquired.

4.2: Technologies Used

The following are the tools and technologies used to develop this project:

Technical Field	Technologies Used
Front End	React Js, HTML, CSS, JSX
Back End	Node Js
Database	MySQL
IDE	Visual Studio Code
Browser	Google Chrome, Mozilla Firefox, Mozilla Firefox Quantum
Version Management	Bitbucket, Google Drive
Web Hosting	Heroku: Cloud Application Platform
Collaboration Tools	TeamViewer 13, Discord
Documentation	Google Docs, Creately
OS	Linux Mint version 18.2 +

4.3: Constraints of Use

The following explains the constraints of the system under development some of which are obvious design decisions, others are regarding the tools we used during development or because of lack of resources.

- Because the software is a web based application, it requires an web browser capable of running JavaScript.
- The application requires internet connection in order to work properly.
- The application, for now uses free web hosting which has its limitations.
- Assembling components together does not remove the sub-components from the database, which is a deliberate choice to preserve integrity and avoid confusion.
- The version one of the system will be used locally and therefore it does not contain a login module, different users and their privileges are not clearly stated in the specification.

4.4: System Architecture

The following section describes the architecture of the system backend, which comprises of the MySQL database that acts as the data repository and the node server which accepts requests from clients and sends those requests to the database, then receives response from the database and send the response back to the client. The clients are instances of the web application, running on client side. Note that the database and the node server are hosted on separate servers, hereafter we will refer to the node server as 'Server' and database server as 'Database'.

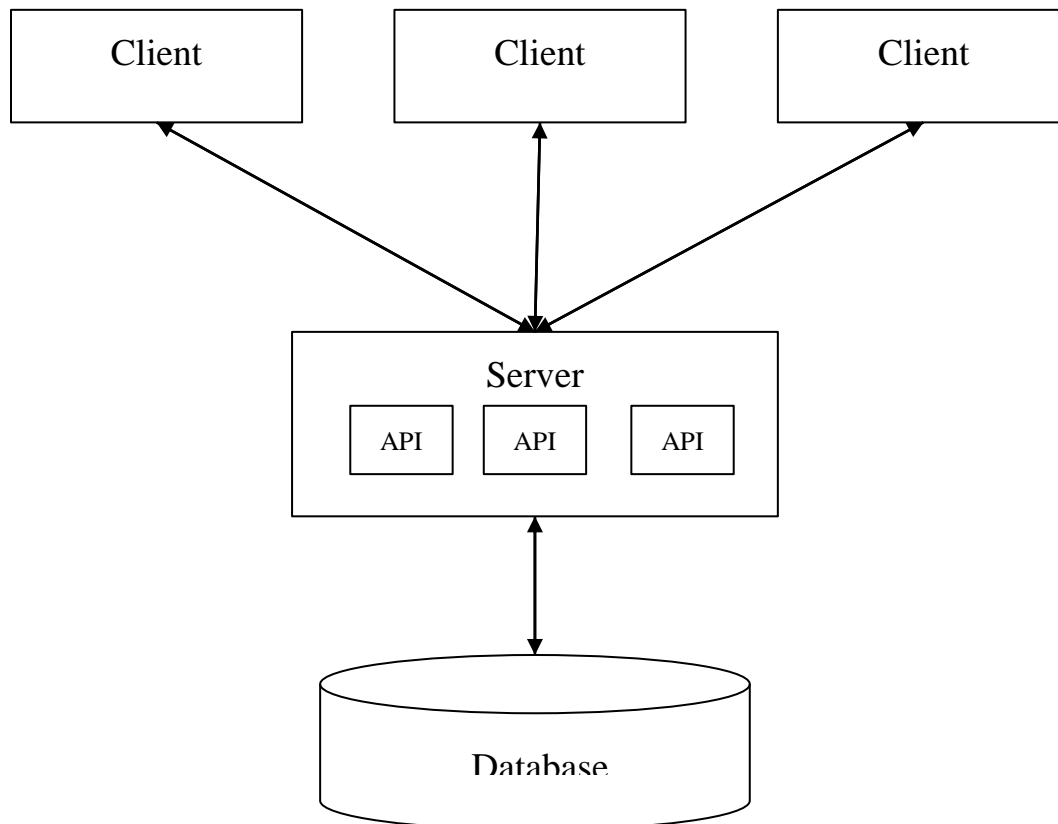


Figure: Interaction between the Clients, the Server and the Database

4.4.1: Database Architecture

The database houses all the necessary data of the Rental Inventory Management System. In brief these includes, customer information, inventory asset information, order and asset configuration information among others. Because the database plays such a crucial and central role in the system architecture we decided to utilize MySQL DBMS, which provides all the necessary security, ease of access and fast response all the while satisfying the ACID properties that are necessary for a safe, secure and recoverable system. Furthermore the database is hosted in a separate server which is not related to the node server.

The following tables show the descriptions of each entities in the database:

Entity name: Asset

Asset table stores the individual assets in the database, along with all their static data, like make, purchase date, part code etc. The dynamic attributes, i.e, the attributes that are dependent on the type of the asset (asset-type) are stored in the asset_details table.

Attribute	Data Type	Constraints
id	int(11)	primary key, auto_increment
asset_type_id	int(11)	not null, foreign key references Asset_Type.id
serial_no	varchar(100)	unique
purchase_date	timestamp	
purchase_price	decimal(10,2)	
supplier	int(11)	
warehouse_location	int(11)	
procurement_date	timestamp	
status	varchar(50)	
create_timestamp	timestamp	
update_timestamp	timestamp	
part_code	varchar(50)	
make	varchar(100)	
warranty_end_date	timestamp	
transfer_order_no	varchar(50)	
comments	varchar(100)	
supplier_invoice	varchar(100)	
supplier_date	timestamp	

branch	varchar(45)	not null
transfer_order_date	timestamp	

Entity name: Asset_Config

Asset_Config table is designed to contain all the historical record of asset configuration. This is associated with the asset modification aspect of the software, i.e., the asset addition or removal of asset. For example we can remove a RAM asset from a Desktop asset and put another RAM asset in that desktop. The child_asset_id specifies where this asset is initially detached from and the parent_asset_id holds the asset_id of the asset it is currently attached to.

Attribute	Data Type	Constraints
id	int(11)	primary_key, auto_increment
asset_id	int(11)	not_null
child_asset_id	int(11)	foreign_key
create_timestamp	timestamp	not_null
update_timestamp	timestamp	not_null
parent_asset_id	int(11)	foreign_key
status	int(11)	not_null

Entity name: Asset_Details

Asset_Details contains all the dynamic attribute values of all the assets. We get the attribute_id from the Asset_Types_Attributes table (that contains the record of all dynamic attribute names) and asset_id from Asset table (which contains all of the assets and their static attributes) and using these two foreign keys we can uniquely identify the attribute value of a particular asset. This mechanism allows the user to dynamically define any asset type and start storing data values of that asset.

Attribute	Data Type	Constraints
id	int(11)	primary_key, auto_increment
asset_id	int(11)	foreign_key references Asset

attribute_id	int(11)	foreign_key references Asset_Types_Attributes
attribute_value	varchar(500)	
create_timestamp	timestamp	not_null
update_timestamp	timestamp	not_null

Entity name: Asset_Types

Asset_types table contains the names of all the asset types that exists in the database. All the assets in the Asset table must belong to one asset type. These asset types are defined by the user. The names of the dynamic attributes that belong to an asset type are recorded in the Asset_Types_Attributes table.

Attribute	Data Type	Constraints
id	int(11)	Primary_key, auto_increment
type_name	varchar(100)	not_null
is_active	tinyint(4)	
create_timestamp	timestamp	
update_timestamp	timestamp	

Entity name: Asset_Type_Attributes

Asset_Type_Attributes contain all the (dynamic) attribute names belonging to the Asset_Types. Notice that this table contains the attribute names, not the attribute values for an asset, that resides in the Asset_Details table.

Attribute	Data Type	Constraints
id	int(11)	primary_key auto_increment
asset_type_id	int(11)	not_null , foreign key references Asset_Type
attr_name	varchar(100)	

is_modifiable	tinyint(4)	
is_mandatory	tinyint(4)	
is_active	tinyint(4)	
create_timestamp	timestamp	not_null
update_timestamp	timestamp	not_null

Entity name: Challan_Draft

This table is used to keep challans as drafts, which are saved by the user, and not submitted. The records in this table specify the type of the challan (e.g. rental), challan_description, that acts as the label of that draft and finally the challan data is stored in challan_details in JSON format, which can later be retrieved by the application to restore the draft, then the user can resume the fill up process.

Attribute	Data Type	Constraints
id	int(11)	primary_key, auto_increment
challan_type	varchar(55)	
challan_description	varchar(100)	
challan_details	longtext	
create_timestamp	timestamp	not_null
update_timestamp	timestamp	not_null

Entity name: Customer

Customer table holds the current name and previous names of the customer along with their PAN number. For the location and contact details of the customer refer to the Customer_Location_Master table. Each tuple in here represents one customer. Customer in this case can range from a large company to an individual.

Attribute	Data Type	Constraints
Customer_Id	int(11)	Primary_key, auto_increment

CName	varchar(100)	not_null
updated_date	timestamp	not_null
created_date	timestamp	not_null
Previously_Known_As	varchar(100)	
Pan_No	varchar(45)	not_null
Comments	varchar(200)	

Entity name: Customer_Location_Master

Customer_Location_Master table holds the contact details, location addresses, etc of all the customers. Each tuple in this entity set represents an address, where each address belongs to only one customer. Four contact details (contact name, number, email) can be associated with each address, where providing one set of the contact detail is mandatory. One address each belonging to a unique customer must be declared main (isMain: true) to signify that is the main address of the customer.

Attribute	Data Type	Constraints
CID	int(11)	Primary_key, auto_increment
Customer_Id	int(11)	Not_null, foreign key references Customer
Address	varchar(200)	not_null
GST_Value	varchar(45)	
Contact_Person_1	varchar(45)	not_null
Contact_Number_1	varchar(20)	not_null
Email_1	varchar(45)	not_null
Contact_Person_1_Valid	tinyint(4)	
Contact_Person_2	varchar(45)	
Contact_Number_2	varchar(20)	
Email_2	varchar(45)	
Contact_Person_2_Valid	tinyint(4)	

Contact_Person_3	varchar(45)	
Contact_Number_3	varchar(20)	
Email_3	varchar(45)	
Contact_Person_3_Valid	tinyint(4)	
Contact_Person_4	varchar(45)	
Contact_Number_4	varchar(20)	
Email_4	varchar(45)	
Contact_Person_4_Valid	tinyint(4)	
Is_Main	tinyint(4)	not_null
Is_Valid	tinyint(4)	not_null
created_date	timestamp	not_null
updated_date	timestamp	not_null
SEZ	tinyint(4)	not_null
City	varchar(45)	
State	varchar(45)	
Pincode	varchar(45)	

Entity name: Order_Master

Order_Master table stores the details of each order. This includes the customer id (that bought / rented it), total amount of money charged, order data etc. Each tuple in this relation is an order, which may include multiple assets, details of which are stored in Order_Details.

Attribute	Data Type	Constraints
ID	int(11)	Primary_key, auto_increment
customer_id	int(11)	not_null , foreign key references Customer
total_amount	double	

challan_number	int(10)	not_null
order_date	timestamp	
customer_location_id	int(11)	not_null
parent_challan_id	int(11)	
eway_number	varchar(45)	
po	varchar(45)	
po_reference	varchar(45)	
cn_number	varchar(45)	
delivery_person_name	varchar(50)	
comment	longtext	

Entity name: Order_Detail

Order_Detail table holds the details of each order, i.e, what assets belong in that order, what was the daily unit price of each asset, what was the rental period of each order, etc.

Attribute	Data Type	Constraints
oid	int(11)	Primary_key, auto_increment
order_id	int(11)	not_null
asset_id	int(11)	Not_null, foreign key references Asset
rental_begin_date	timestamp	
rental_end_date	timestamp	
daily_unit_price	double	
current_procurement_price	double	
total_unit_price	double	
gst_value	double	
total_value	double	

status	tinyint(4)	not_null
--------	------------	----------

4.4.2: Server Architecture

The server is the entity that provides connections between the client application and the DBMS. The Server is a node js application consisting of multiple APIs which enables the interactions among the application and the database. Multiple instances of client applications interact with one server. The server is hosted online on Heroku server.

The following is a brief list of the commonly used APIs that are defined in the server and invoked when the client request the server to do so.

Name	/get_asset
Method	GET
Input	-
Output	List of all asset type names (JSON)
Description	Get all asset type names

Name	/get_asset_type
Method	GET
Input	Type_name :String
Output	List of all attribute names of that asset type (JSON)
Description	Fetch attributes by asset type names

Name	/insert_asset_value
Method	POST
Input	Asset_type_name : String Static_data : JSON

	Dynamic_data : JSON
Output	isSuccess (JSON)
Description	Create new asset instance, save the static and dynamic as attributes of the asset.

Name	/insert_asset_type
Method	POST
Input	assetTypeName : String assetTypeAttributes : JSON
Output	Success (JSON)
Description	Create new asset type.

Name	/insert_customer
Method	POST
Input	Customer_name : String pan_number : String comments : String
Output	Success (JSON)
Description	Insert a new customer into database.

Name	/get_asset_status_count
Method	GET
Input	
Output	List of out_of_stock, in_stock and damaged assets (JSON)
Description	Get count of all the assets that are in and out of stock

Name	/get_customer
-------------	---------------

Method	GET
Input	Customer_id : String
Output	isSuccess, customerDetails and locationDetails (JSON)
Description	Get all the customers details.

Name	/get_asset_type_customer_name
Method	GET
Input	Asset_type_id : String
Output	List of customer name (that has the asset) and asset details : JSON
Description	Get the details of all the assets of this asset type along with their customer (when rented)

Name	/in_damaged_stock
Method	GET
Input	Status : String Id : String
Output	Array of asset details : JSON
Description	Get the details of all the assets of this asset type with status instock(1) or damaged(2)

Name	/get_all_values
Method	GET
Input	Type_name : String
Output	Static and dynamic attribute values : JSON
Description	Get the static and dynamic attributes of all the assets having the specified asset type name

Name	/change_inventory_status
-------------	--------------------------

Method	GET
Input	Array of asset_id : JSON
Output	Success : JSON
Description	After renting changes inventory status to unavailable

Name	/change_config_table_on_add
Method	POST
Input	List of cart items : JSON
Output	Success : JSON
Description	When configuring assets by adding them, saves the parent child data on config table.

Name	/order_create
Method	POST
Input	List of order details : JSON
Output	Success : JSON
Description	Create an entry into order table according to the data.

Name	/get_all_modifiable_values
Method	GET
Input	Type_name : String
Output	Attribute values of assets (static & dynamic) : JSON
Description	Fetches only those assets which are modifiable

Name	/remove_asset_value
Method	GET
Input	Asset_id : String

	attribute_id : String Attribute_value : JSON
Output	Success : JSON
Description	Removes attributes from an asset. For example, removing 4GB RAM from an asset of Desktop type

Name	/change_config_table_on_delete
Method	POST
Input	Serial_no : String id : String
Output	Success : JSON
Description	When attribute is removed from asset, this api reflects the changes in config table.

Name	/return_from_repair
Method	GET
Input	assetId : String
Output	Success : JSON
Description	When assets come back from repair, set the status of the specified asset from 'damaged' to 'normal'

Name	/change_status_on_return
Method	GET
Input	assetUpdates : JSON orderUpdates : JSON
Output	Success : JSON
Description	When assets are returned to the inventory, change the database to reflect that they are now available.

Name	/modify_customer
Method	POST
Input	customerId : String customer_name : String comments : String
Output	Success : JSON
Description	Fetches all damaged assets from database

Name	/get_asset_config
Method	GET
Input	-
Output	Asset_config_id : JSON Asset_config_child : JSON
Description	Get asset configuration details of all the assets

Name	/get_customer_order_details
Method	GET
Input	Customer_id : String
Output	Success : JSON
Description	This API is used to retrieve the customer's order details from the order_detail table by searching using customer_id.

Name	/change_config_status
Method	GET
Input	Id : String
Output	Success : JSON
Description	This API is used to change the config status of an assembled asset after it

	has been removed from the concerned asset.
--	--

Name	/send_for_repair
Method	GET
Input	Id : String
Output	Success : JSON
Description	This API is used to mark assets as 'sent for repair', therefore they will not appear as candidates for rental process.

Name	/modify_asset_type
Method	POST
Input	Type_name : String Attributes : JSON
Output	Success : JSON
Description	API to insert more attributes into existing asset-type

Name	/insert_challan_draft
Method	POST
Input	challanType : String challanDescription : String challanDetails : JSON
Output	Success : JSON
Description	Saves the challan state to database in JSON format, where it is stored as drafts, it can be recovered later and resumed.

Name	/get_challan_drafts
Method	GET
Input	-

Output	Success : JSON
Description	Get the list of all of the challans that are saved as drafts, this does not include the challan details.

Name	/get_challan_details
Method	GET
Input	challanId : String
Output	Challan detail : JSON
Description	API that accepts a challanId and retrieves the challan details of that draft.

4.5: System Operations

In this section of the project documentation, a walkthrough of all system operations is provided. The system has been designed as a website with multiple pages and components. The various descriptions of the pages and their operations are given below.

JSX	Type	Name	Functionality/Operation
AddAddress.jsx	Page		This page is utilised to add a new address for the customer. As specified by the customer's requirements, no address will be overwritten or modified. Hence if an address becomes invalid, it will be marked so and in its place a new address will be added if required.
AddAsset.jsx	Page	Add to Inventory	This page is utilised to add a new asset to the inventory. We start by selecting an asset type and then enter relevant asset information. Facility is provided to enter multiple serial numbers for different assets with same specifications. Facilities are also provided for scanning serial numbers for

			assets instead of manually entering them. Validation is provided with appropriate error messages to inform the user what has gone wrong while entering asset details.
AddAssetType.jsx	Page		This page is utilised to dynamically create different categories of assets to properly group assets in the inventory.
AssetAddition.jsx	Page		This page is utilised to modify an existing asset which is with the customer with another asset which can be both in-stock or with the customer. So if the customer has a Desktop which contains a damaged RAM, another RAM can be sent to the customer for modification or the customer can simply use a RAM which has already been rented to him to modify his asset.
ChallanDraft.jsx	Page	Saved Challan Drafts	This page is utilised to show existing challan drafts which has not been submitted yet and can still be modified before final submission. This provides further flexibility of service as in many cases the delivery person name or the PO reference etc are not known till later. From this page we can choose an existing challan and update it and submit it when all work is done.
Dashboard.jsx	Page	Dashboard	This is the 'homepage' of the website and is used by the office staff to show reports, daily activities and various other tabular information. It contains all the information about assets, orders, customers, configuration history and inventory data.
DisplayAssets.jsx	Page	Order Asset	This page is the first page in the entire order process. It is utilised to show a list of assets chosen by their category. We can filter then using the multitude of filter provided for the user's benefit. Multiple assets are chosen and then added to cart for further assembly processes. The page/component following the DisplayAssets is Checkout.
EditAssetType.jsx	Page	Edit Asset Types	This page is used to modify an existing category of asset to add further

			characteristics. It has been done in such a way that data about that category's assets already in the inventory are not hampered.
GenerateChallan.jsx	Page		This is the third step in the ordering process. It involves creating the entire customer invoice and either submitting it to create challan or saving it as challan draft. The user is capable of selecting a challan type, selecting customer, his address, contact information and then creating the order by entering rental begin date, unit price etc.
InsertCustomer.jsx	Page	Add Customer	This page is used to enter the customer information of the client company's customers. Name, Pan No and dynamic address insertion is possible. Multiple addresses can be entered while provisions have been provided to enter multiple contact person information. Validation is provided on all fields so as to ensure no erroneous data is entered.
ManageCustomer.jsx	Page	Manage Customer	This page is utilised to edit the various customer details.
RemoveAsset.jsx	Page		This page is utilised to remove asset attributes. For example if an asset Desktop has a 4GB RAM which needed to be replaced or taken out we can utilise this page to remove that attribute. This asset attribute can then be added into the existing inventory as a RAM.
ReturnAssets.jsx	Page	Manage Inventory	This page is used to manage the inventory by processing the returned assets from the customer, checking them for damages and appropriately detailing them and also configuring assembled assets.
Challan.jsx	Component		This component is used to render the challan for print out. The challan is designed in accordance with the manual challans used by the client company.
Header.jsx	Component		This component is used to render the blue

			header on top of each page containing the company logo and name.
SideMenu.jsx	Component		This component is used to render the side-menu which contains the quick navigation for all pages. It is opened by clicking the hamburger menu on the top left corner of the pages.
AssetCard.jsx	Component		This component is the second step of the ordering process. It is used to render the chosen assets in the form of cards. There they can be chosen for assembly before sending them for rental or other purposes.

Miscellaneous

JS/CSS	Type	Purpose
App.css	CSS	CSS file to design the pages and the associated headers.
App.js	JS	JS file of the application which contains the browserrouter and the switches. Any page link which we want to utilise in our app needs to be added here to be accessed in our code.
index.css	CSS	CSS file to design the side menu and the hamburger menu.
Server (index.js)	JS	Contained as a separate project this file contains the apis written for the backend. The project resides in heroku and has been added as a dependency in the package.json file of our project. The index.js is used to write the apis using

		Node.js.
--	--	----------

5: Requirements Traceability

The software requirements are defined as the descriptions of the various features and functionalities of the proposed system. Requirements convey the expectations of the users for the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view. Before the entire design process begins a document known as SRS (Software Requirements Specification) is created which is basically a collection of all the customer's proposed requirements which are collected in the form of a comprehensive and concise document. As with all projects, this project too contains an SRS. The functional and non-functional requirements from the SRS document are provided here.

Functional Requirements:

Functional Requirements are calculations, technical information, processed data and functionalities that define what the system is supposed to or is going to accomplish. Following are the functional requirements for the proposed system.

- **Interface Requirements:** Properly detailed screens maintaining a professional profile of a page without exhaustive designs so as to maintain the decorum of office space while maintaining an user friendly approach, easily grasped by the users. Large use of autocomplete fields are done so as to make the work of the office staff easier, who will deal in serial numbers of 1000s of assets. The order page includes multiple filters to make searching for multiple data simultaneously easier. Large scale use of Loaders ensure that data transaction maintains a graphical interface. Validation of fields is provided with proper labels to guide the user when they go wrong. Extensive use of date-pickers enable the users to work with date type data easily. Also react-print is used to interface with external printer device for easy print of challan form.
- **Business Requirements:** Data validation enables that only valid and constrained data will be transferred using the backend. Transaction blocks are heavily imposed to maintain the ACID properties of database transactions so that huge amount of data transfer isn't made a slave to faulty internet connections leading to frequent inconsistent data. The frontend (React.js) and the backend (Node.js) will work asynchronously ensuring fast data transfer.
- **Regulatory/Compliance Requirements:** The product has been studied extensively and no legal complications have arisen. Data is to be stored in the database at a secure server protecting the various asset and customer information. Each page is secured and data transfer is smooth and safe ensuring that no sensitive data is leaked. The product is made after complying with every company protocol and following every regulation necessary.

- **Risks:** To reduce the circumstances under which the requirements may not be satisfied, all the designers must have an idea of developing websites previously and they must be aware of html restrictions and cross browser implementations before starting the design. In order to reduce the probability of this occurrence, the entire design team must have basic knowledge of React.js, Node.js and MySQL in general and JavaScript in particular.

- **Technical Issues:** In order to satisfy this requirement design should be simple and all the different interfaces should follow a standard template. A proper standard is maintained for colours and design so that switching between web pages is a relatively user friendly experience.

Non Functional Requirements:

As opposed to functional requirements, non functional requirements are those categories of requirements by which we can judge the operations of the system rather than what the system is supposed to do or rather what the system does. The non functional requirements are closely related to system architecture as opposed to functional requirements that specify the system design. Following are the set of non functional requirements identified for the project.


- **Performance Requirements:** The system provides for fast data transaction while maintaining the ACID property of transactions. Simultaneous access to the system is allowed while ensuring data integrity and consistency. Furthermore the durability and recoverability of the stored data is necessary for data security.

- **Availability and Reliability:** The software system ensures that no data is overwritten under any circumstances. All data is backed up in the database ensuring that all data persists in the database. Any data which is updated will be referenced by the updated timestamp from the database.

- **Security Requirements:** Even though there is an absence of login and registration system, the data is kept fairly secure for intra-office use. The data transfer is done using the highly reliable and asynchronous Node.js ensuring data consistency and transaction following ACID properties. The website, server and database are all hosted in secure servers to ensure full data security. Physical access to the system's MySQL database will be restricted to only a few authorized personnel.

6: User Interface

Dashboard: Containing a brief overview of the current inventory status along with the customer details. The overview can be both customer specific or asset specific.

 **Dashboard**

Asset-Type	In stock	Out of Stock	Damaged	Total
Desktop	11	11	7	29
Ram	1	10	1	12
TOTAL >	12	21	8	41

12
21
8
41

Total Customers: 7

Customer Name	Previous Names	PAN No.
Avilash Kumar	Avilash Chatterjee	as12as
Ritaraj Pal	Ritaraj Dutta	adwe12
Meghnad Das	Meghnad Das	11qwsd
Diganta Das	Diganta Datta	d-123
Amarjit Sen	Amarjit Das	d1234
Ibm Pvt Ltd	Ibm	i123
Batman		batpan123

Config Details

Asset Type	Make	SI No.	Comment	Branch
Desktop	DELL Inspiron	d3	Hi Hi	Pune
Ram	DELL Inspiron	d1	ABC	Pune
	DELL Inspiron	d5	XYZ	Pune
	DELL Inspiron	d2		Pune

Batman

Customer ID: 14
 Name: Batman
 Previous Alias:
 PAN: batpan123

Comments: Gotham Rocks .DC bad
 ,Gotham Rocks .DC bad ,Gotham Rocks .DC
 bad ,Gotham Rocks .DC bad

[Order History](#)
[Close](#)

Report after clicking on ‘out of stock’ item under Ram category, which shows the results as follows.

The screenshot shows the 'Out Stock Details' report in the Rental Inventory Management System. The header includes the 'computerexchange' logo and the text 'RENTAL INVENTORY MANAGEMENT'. Below the header, the report title 'Out Stock Details' is displayed. A tab labeled 'Assets Out of Stock' is active. The main content is a table with the following columns: Asset Name, Serial No., Customer Name, Address, City, State, and PIN Code. The table lists 13 rows of data, including assets like Gigatron, Zebronics, Giga, and Lion, with their respective serial numbers, customer names, addresses, cities, states, and PIN codes.

Asset Name	Serial No.	Customer Name	Address	City	State	PIN Code
Gigatron	r1	Ritaraj Pal	5 Marine Drive	Mumbai	Maharashtra	800000
Zebronics	r2	Ritaraj Pal	5 Marine Drive	Mumbai	Maharashtra	800000
Gigatron	r3	Ritaraj Pal	5 Marine Drive	Mumbai	Maharashtra	800000
Zebronics	r4	Meghnad Das	2/5 we street	Kolkata	West Bengal	700123
Gigatron	r5	Meghnad Das	2/5 we street	Kolkata	West Bengal	700123
Zebronics	r6	Meghnad Das	2/5 we street	Kolkata	West Bengal	700123
Giga	r-test	Meghnad Das	2/5 we street	Kolkata	West Bengal	700123
Zebronics	r4	Avilash Kumar	b 25 park street	Kolkata	West Bengal	700090
Giga	r-test	Avilash Kumar	b 25 park street	Kolkata	West Bengal	700090
Lion	r-test3	Batman	batman 3	gotham	wb	123098
Tiger	r-test2	Batman	batman 3	gotham	wb	123098

Add a new customer to the database which includes their name and pan number along with multiple address information.

Add Customer

Name *
Name

PanNumber *
PanNumber

Comments
Add Comments...

Address #1
Enter Address

City **State** **Pincode**

Main Address SEZ


GST
GST

Main Contact Person **Main Contact Number** **Main Email**

Alternate Contact Information

Contact Person	Contact Number	Email
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>

Panel to add or edit an existing customer's details such as adding new addresses.

RENTAL INVENTORY MANAGEMENT

Select Customer

Batman

Customer Details

Name	Pan Number(Read Only)	Previous Name(Read Only)	Comments
Batman	batpan123		

Customer Address Details

Address(Read Only)	City(Read Only)	State(Read Only)	Pincode(Read Only)	GST Value	Main Contact Person	Main Contact Number	Main Email
batman 1	Gotham	WB	123456	b111	robin	1234560932	bat@yahoo.bi
batman 2	star	wb	111222	b222	super	11220946	b1@yahoo.sp
batman 3	gotham	wb	123098	b333	arrow	73643873	bg@yahoo.ar
batman 4	ggbat	gg	123997	b444	lantern	35546834	su@yahoo.in

Add an asset to the inventory specified by their asset type. The Dynamic attributes vary according to asset's type and the static attributes are the same for each asset.

Dynamic attributes

HDD	<input type="text" value="Enter HDD"/>
RAM	<input type="text" value="Enter RAM"/>
RAM	<input type="text" value="Enter RAM"/>
Motherboard	<input type="text" value="Enter Motherboard"/>
Processor	<input type="text" value="Enter Processor"/>
SMPS	<input type="text" value="Enter SMPS"/>
Sound Card	<input type="text" value="Enter Sound Card"/>
Graphics Card	<input type="text" value="Enter Graphics Card"/>
Server	<input type="text" value="Enter Server"/>

Add new Asset

Select Asset Type:

Serial Number(s)

Quantity:

Branch	Purchase Date	Transfer Order Date	Purchase Price
<input type="text" value="Bangalore"/>	<input type="text" value="05/04/2018"/>	<input type="text" value="05/04/2018"/>	<input type="text" value="125400"/>

Warehouse Location

Procurement Date	Part Code
<input type="text" value="05/04/2018"/>	<input type="text" value="sse45sd"/>

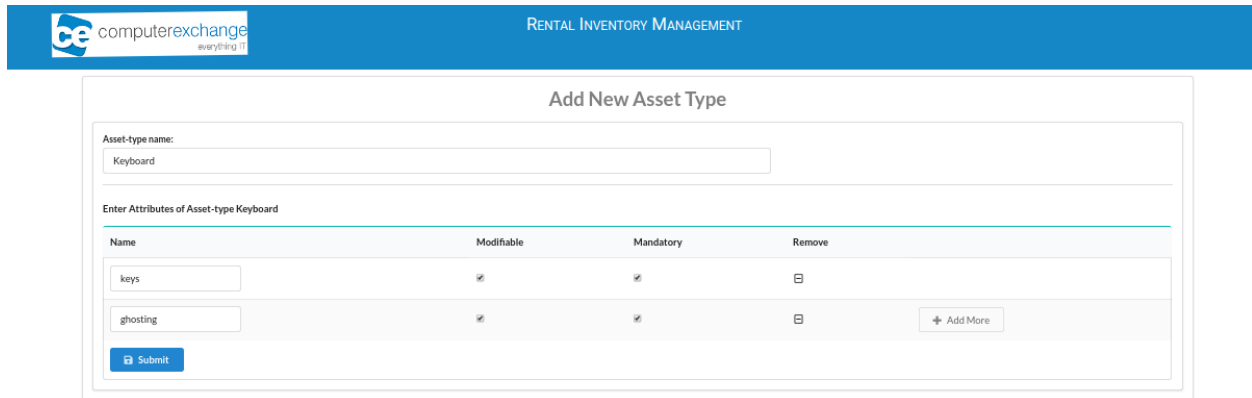
Make	Warranty end date
<input type="text" value="Gigabyte Geforce 180x"/>	<input type="text" value="05/04/2018"/>

Supplier	Supplier invoice no.	Supplier invoice date
<input type="text" value="Supplier"/>	<input type="text" value="Supplier invoice no."/>	<input type="text" value="05/04/2018"/>

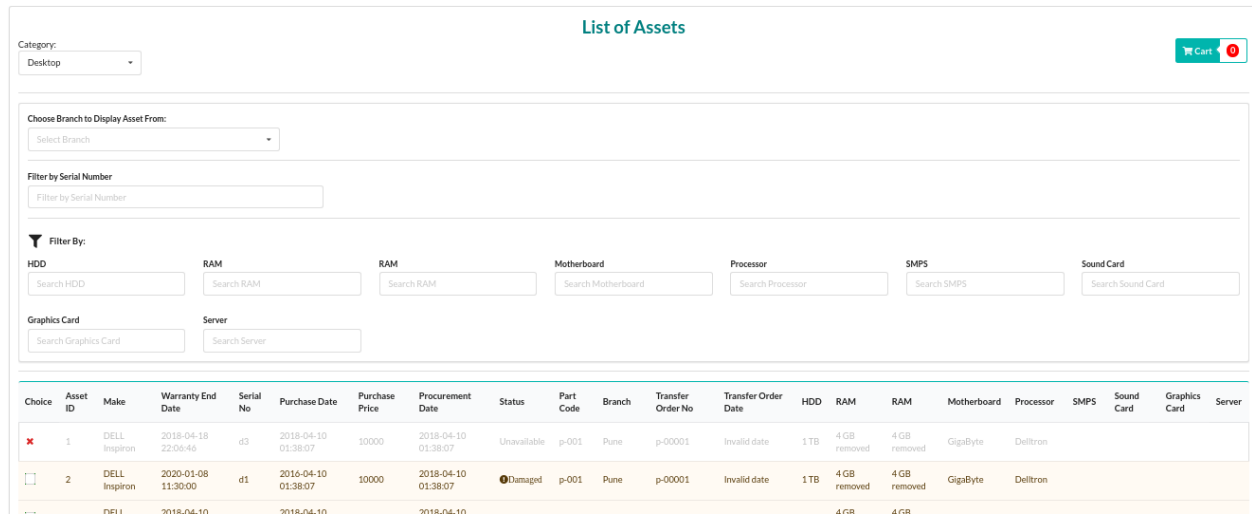
Transfer order

Comment

Define a new asset type under which similar types of assets can be categorised in the inventory.



The order page contains the list of all ‘damaged’, ‘out of stock’ and ‘in stock’ inventory items among which only the ‘in stock’ items are selectable for the next ‘add to cart’ process for generating the challan and invoice entries of aforementioned cart items.



Inventory management page for managing various inventory related tasks, like return of damaged goods, return of verified goods and managing configured goods which are a coalition of various assets or inventory items.

The image displays three sequential screenshots of the Rental Inventory Management System interface. Each screenshot features a blue header with the 'ce computerexchange' logo and the text 'RENTAL INVENTORY MANAGEMENT'. Below the header is a navigation bar with three tabs: 'Manage Returned Assets' (green), 'Manage Modified Components' (blue), and 'Manage Damaged Components' (teal).

The first screenshot shows the 'Manage Returned Assets' tab selected. It includes a search bar for 'Select Customer' and a 'Filter By Serial Number' section with a text input field. A 'Category' dropdown menu is also present, with a red error message below it that reads 'Please enter a valid Asset Type'.

The second screenshot shows the 'Manage Modified Components' tab selected. It features a search bar for 'Select Asset Serial Number' containing the value 'r4'. Below this is a purple button labeled 'Asset Modification Details'. A table displays the following data:

Asset ID	Serial Number	Config Date	Added To	Action
10	r4	2018-04-19 13:25:23	d4	Remove

Take a print out of challan after completing customer invoice.

Mozilla Firefox

about:blank

DELIVERY CHALLAN

COMPUTER EXCHANGE PVT.LTD		www.computerexchangeindia.com Dial: +91-20-65200269/ 09830951013 CIN: 213464245										
#102, First Floor, May Fair Court, Nachiket Park, Dr. Pai Marg, Baner, Pune 411 045												
<p>M/S Batman batman 2, star, wb, 111222</p> <p>Delivery at batman 3, gotham, wb, 123098</p>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;">No.</td> <td>372</td> </tr> <tr> <td>Date</td> <td>04/05/2018</td> </tr> <tr> <td>Order PO#</td> <td>sbi. ef ref</td> </tr> <tr> <td>Contacts</td> <td>arrow</td> </tr> <tr> <td>Messenger</td> <td>John</td> </tr> </table>		No.	372	Date	04/05/2018	Order PO#	sbi. ef ref	Contacts	arrow	Messenger	John
No.	372											
Date	04/05/2018											
Order PO#	sbi. ef ref											
Contacts	arrow											
Messenger	John											
Please receive the goods in good condition and return the challans duly signed												
Qty	Serial No	Description of goods										
1	q6/1 TB/4 GB/8 GB/Gigabyte/AMD/50 Watts/Audiomix/Radeon/20 GG	AMD										
1												
Remarks		Sending on rental basis										
We Have received the goods in good condition.		for COMPUTER EXCHANGE PVT.LTD.										
Customer Signature	Date	Authorised Signature										

Validation on insertion fields.

Address #1

Enter Address

Please enter a valid Address

City State Pincode

Enter City Enter State Enter Pincode

Please enter a valid pincode

Main Address SEZ

GST

GST

Please enter a valid GST or enter NA

Main Contact Person Main Contact Number Main Email

Enter Contact Person Enter Contact Number Enter Email

Please enter a valid Contact Person Name Please enter a valid Contact Number Please enter a valid Email

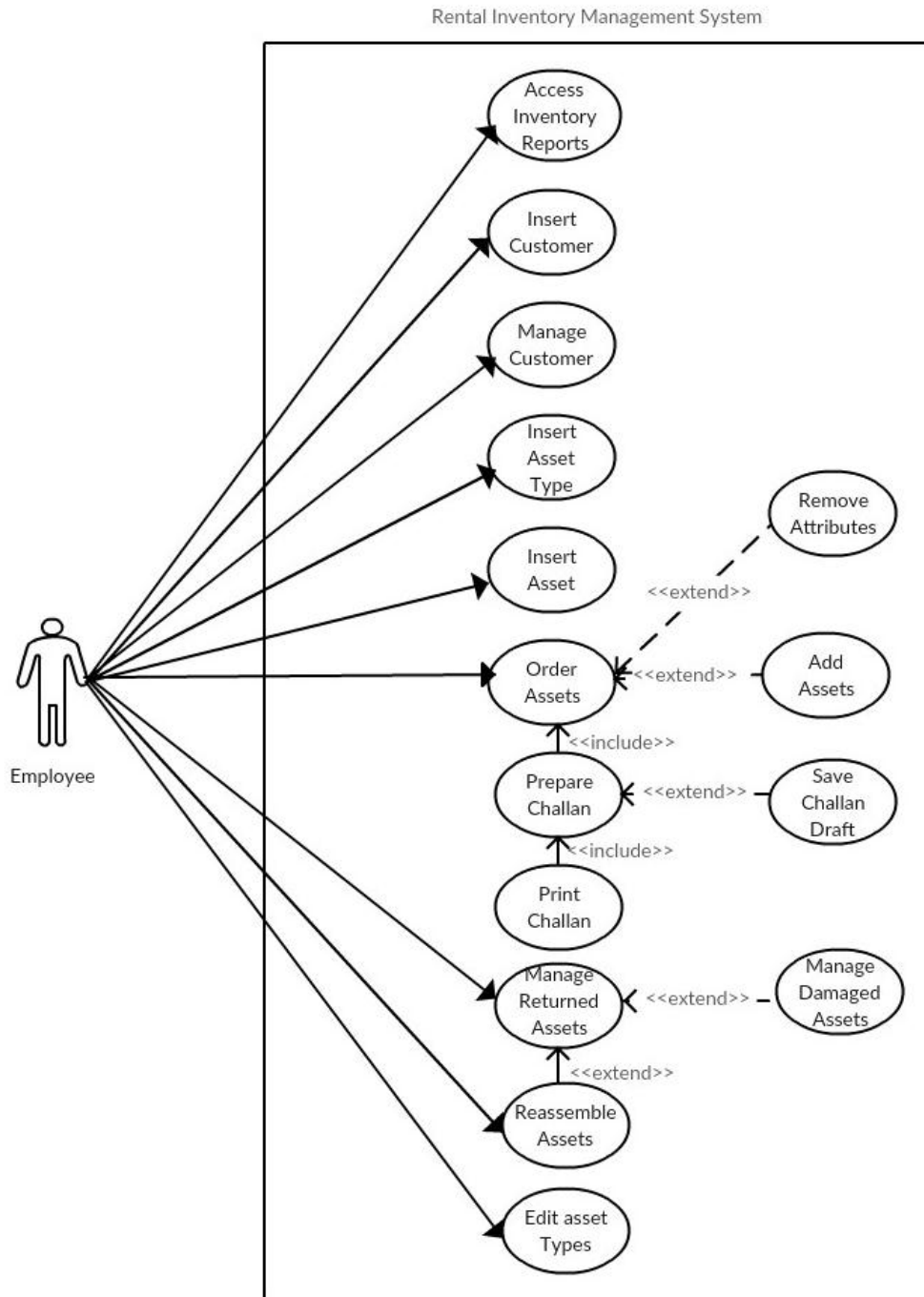
Alternate Contact Information

Contact Person	Contact Number	Email
Enter Contact Person	Enter Contact Number	Enter Email
Enter Contact Person	Enter Contact Number	Enter Email
Enter Contact Person	Enter Contact Number	Enter Email

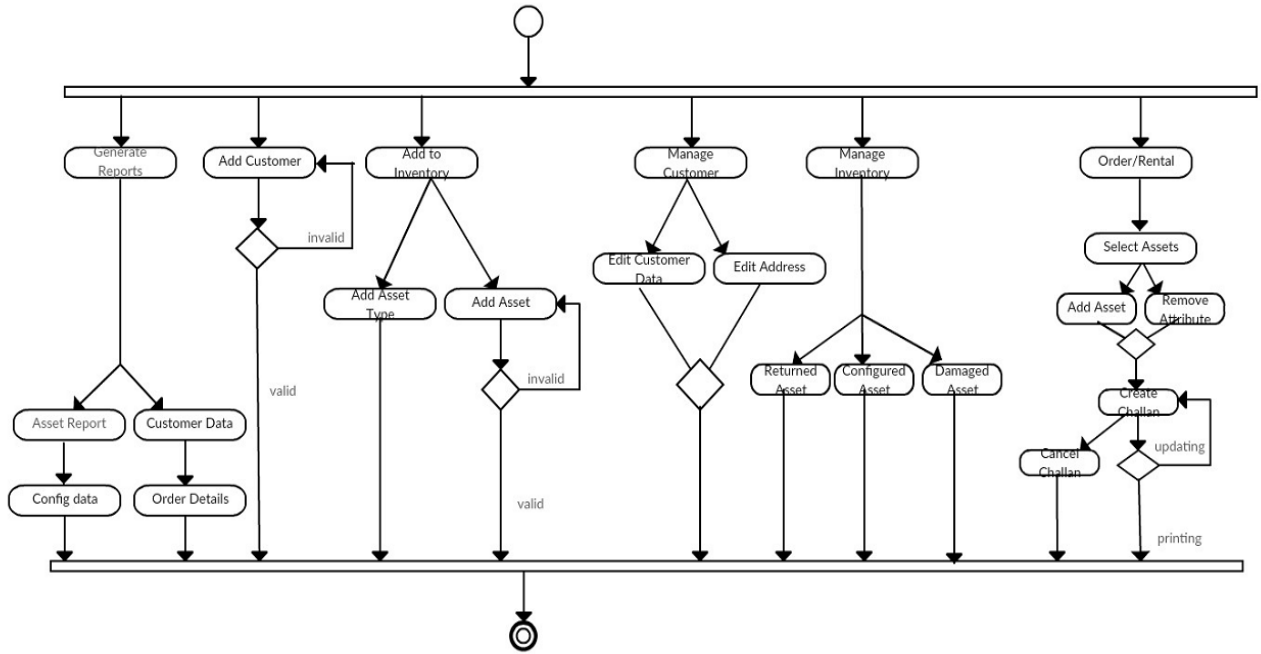
7: Data Model and Storage

7.1: UML diagrams

7.1.1: Use Case Diagram



7.1.2: Activity Diagram



7.2: Functional Diagrams and Algorithm

7.2.1: React Component tree

The following is a simplified representation of the project in a tree structure where each node represents a React component.

The root of the tree is the app component which is the main component that is rendered by the application. This component utilizes three other components: Header (which shows the common header of all the pages), Side menu (which is used for navigation and it can be called by clicking on the proper button on the header), and finally the BrowserRouter which router to all the web pages.

The children to the BrowserRouter are all the web pages that the BrowserRouter can route to.

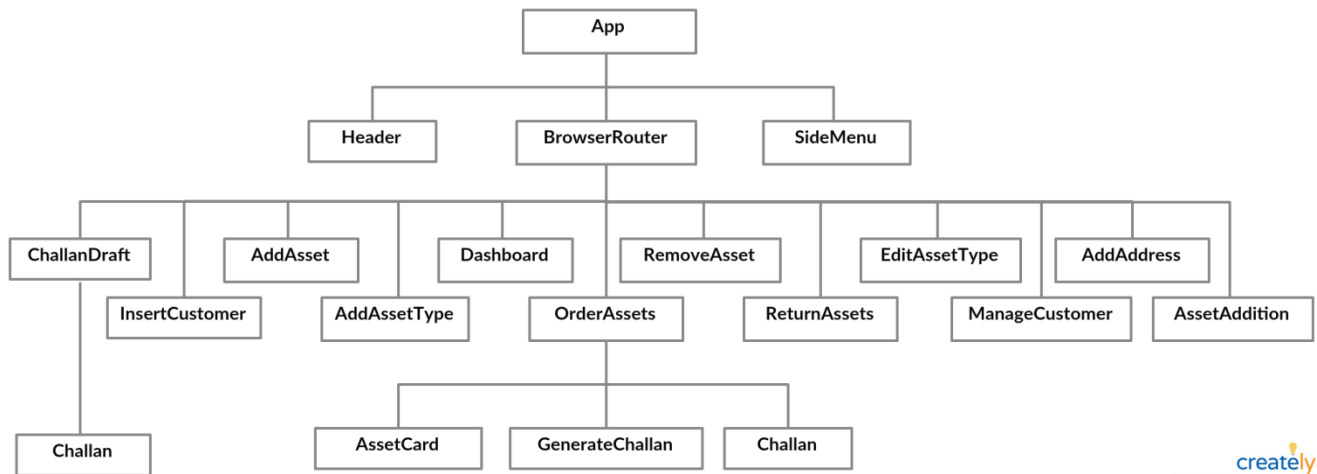
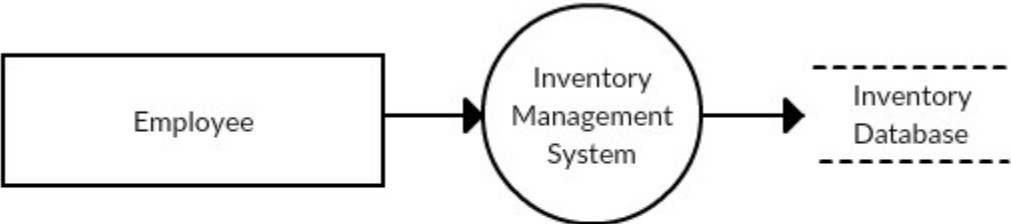


Figure: React Component Tree Structure

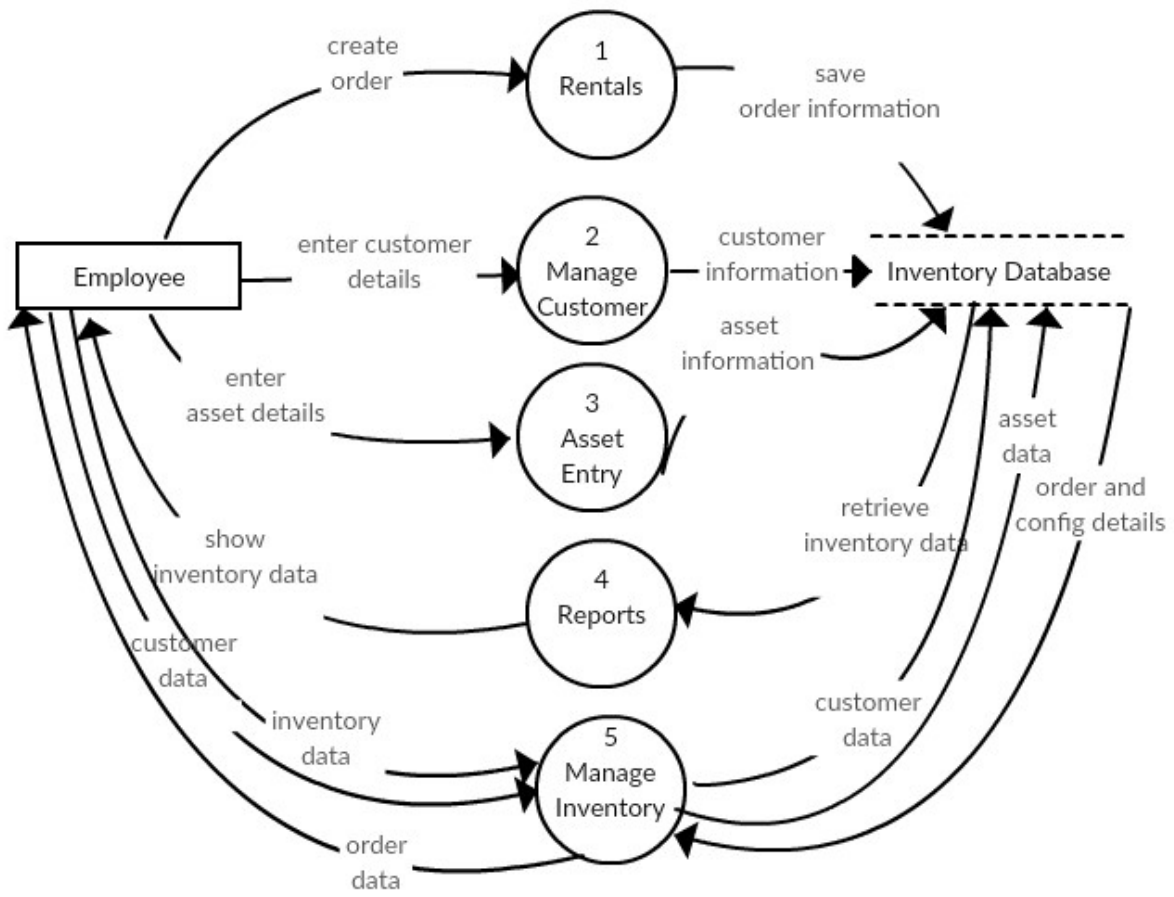
These pages include Dashboard, AddAset, ManageCustomer, InsertCustomer among others. Now, each of these web pages, which are React components utilize other sub components as well. For example OrderAsset component utilizes the sub components: AssetCard to display cart items, GenerateChallan to show challan form and Challan to display the challan and print it.

7.2.2: Data Flow Diagram

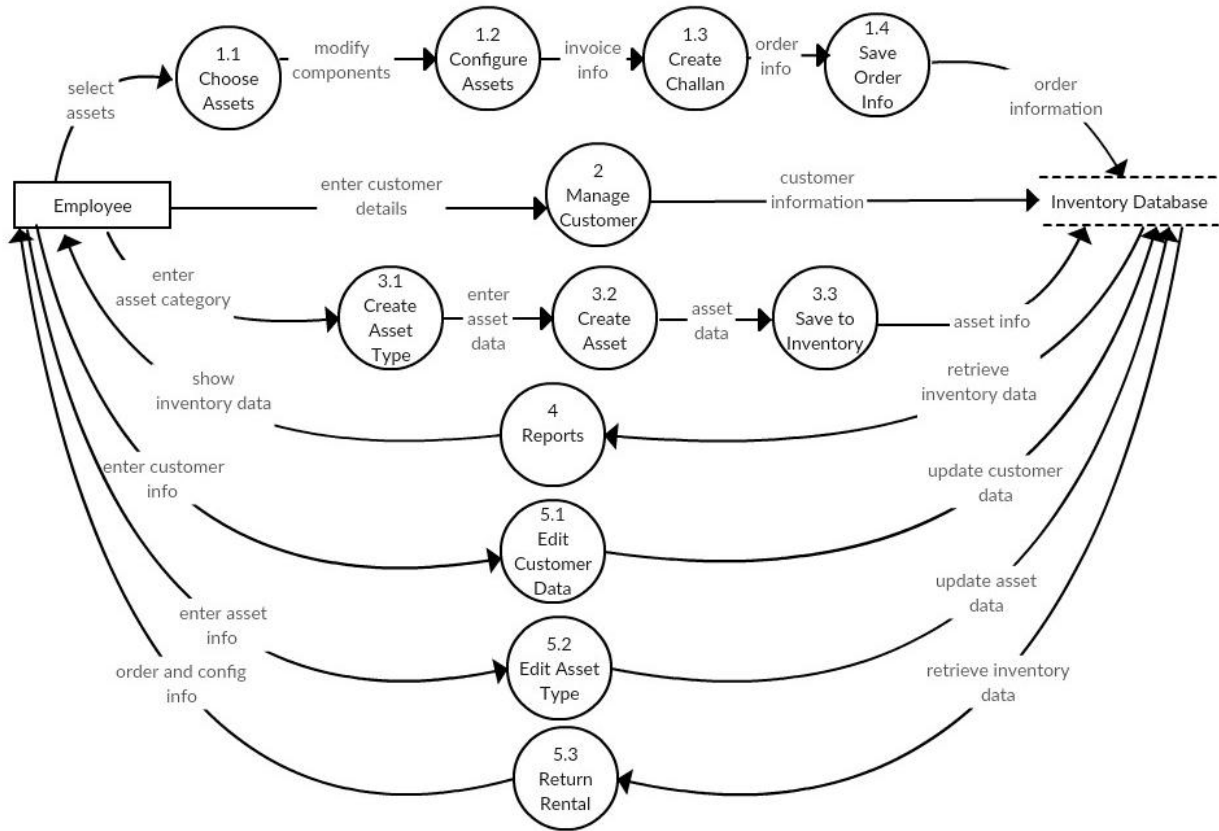
Level-0 (Context Level Diagram)



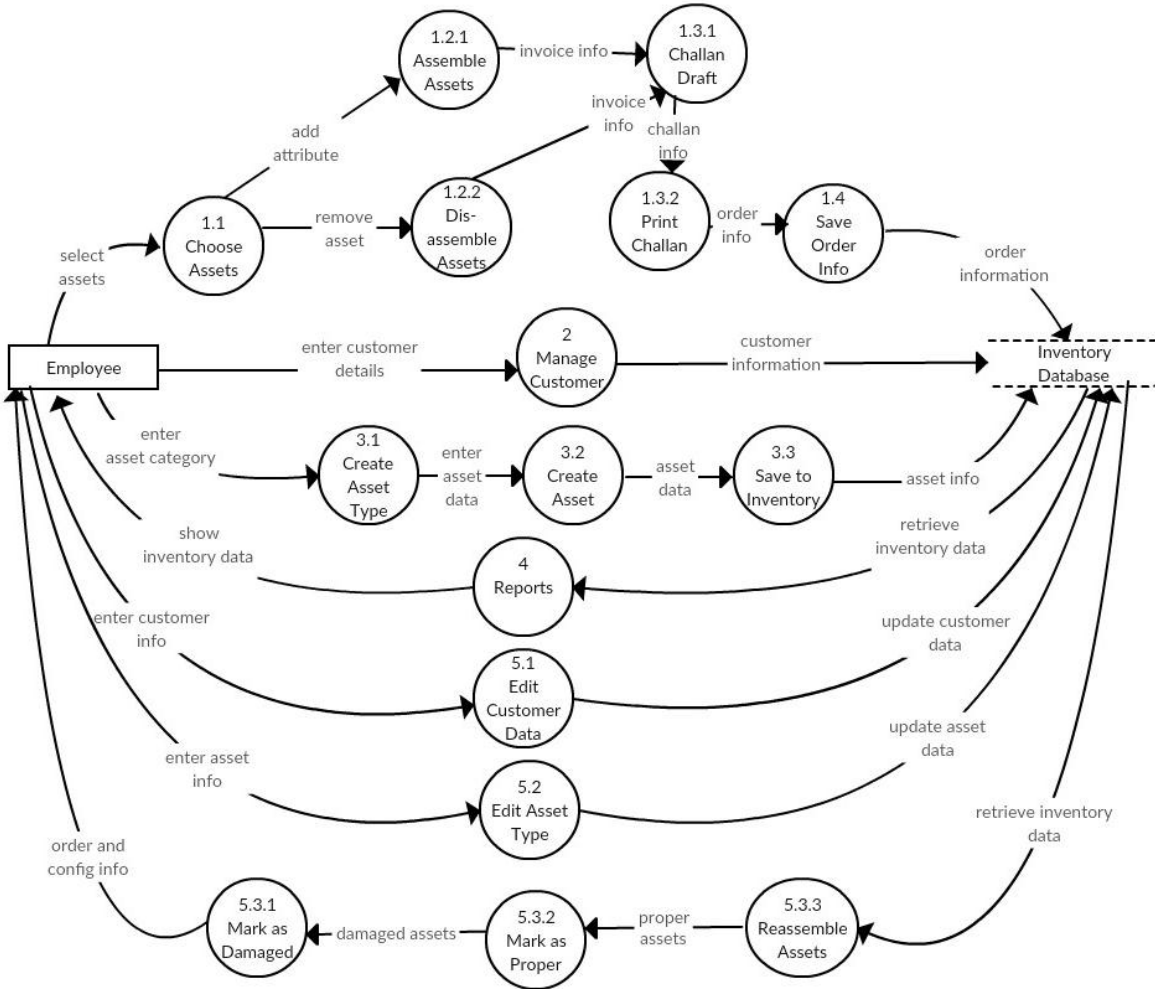
Level-1



Level-2



Level-3



7.2.3: E-R Diagram

The following is the entity relationship diagram of the Rental Inventory Management System database. The entities are described in detail on section 4.1: Database Architecture. This ER diagram illustrates the relationships between the entities.

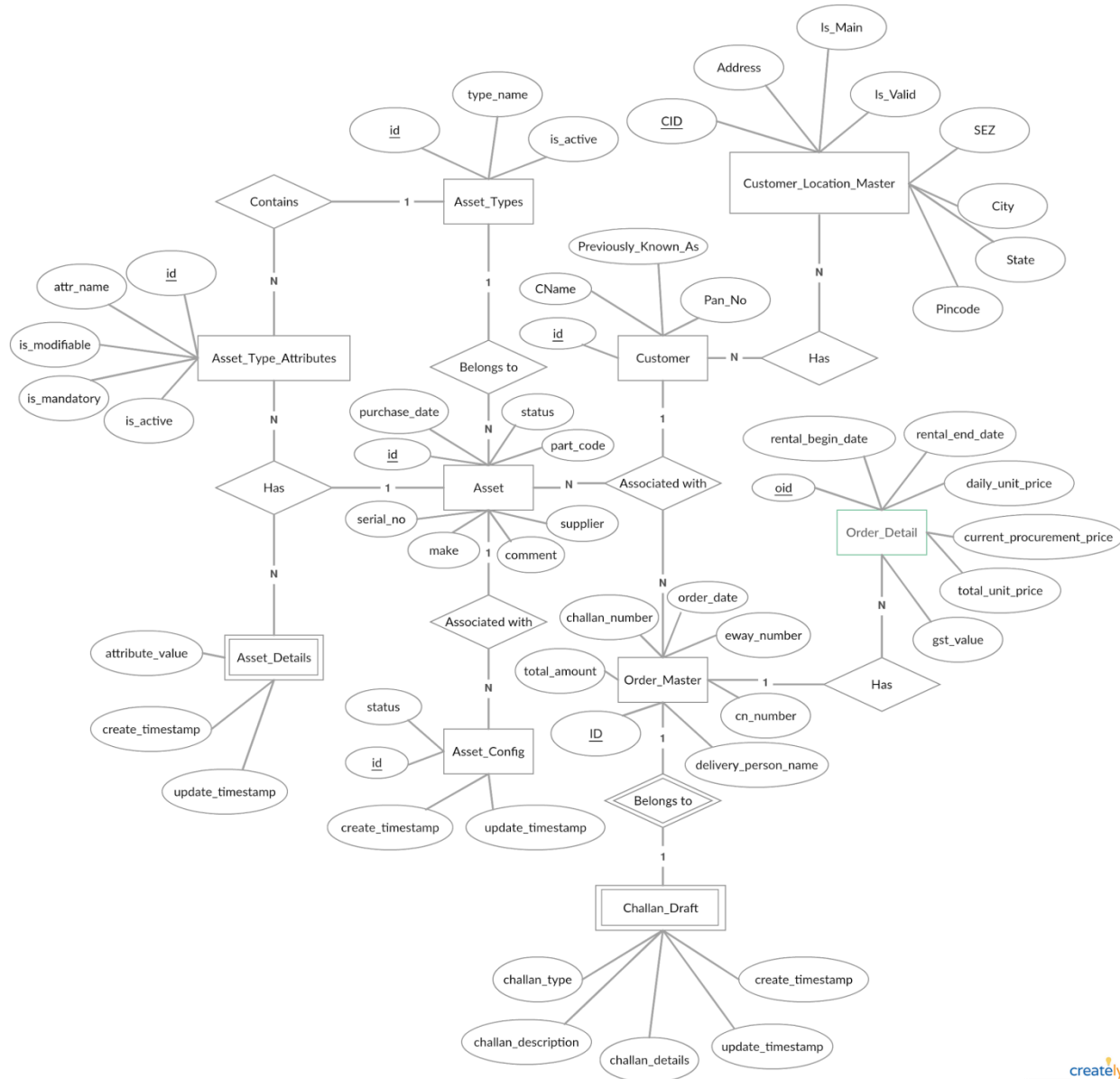
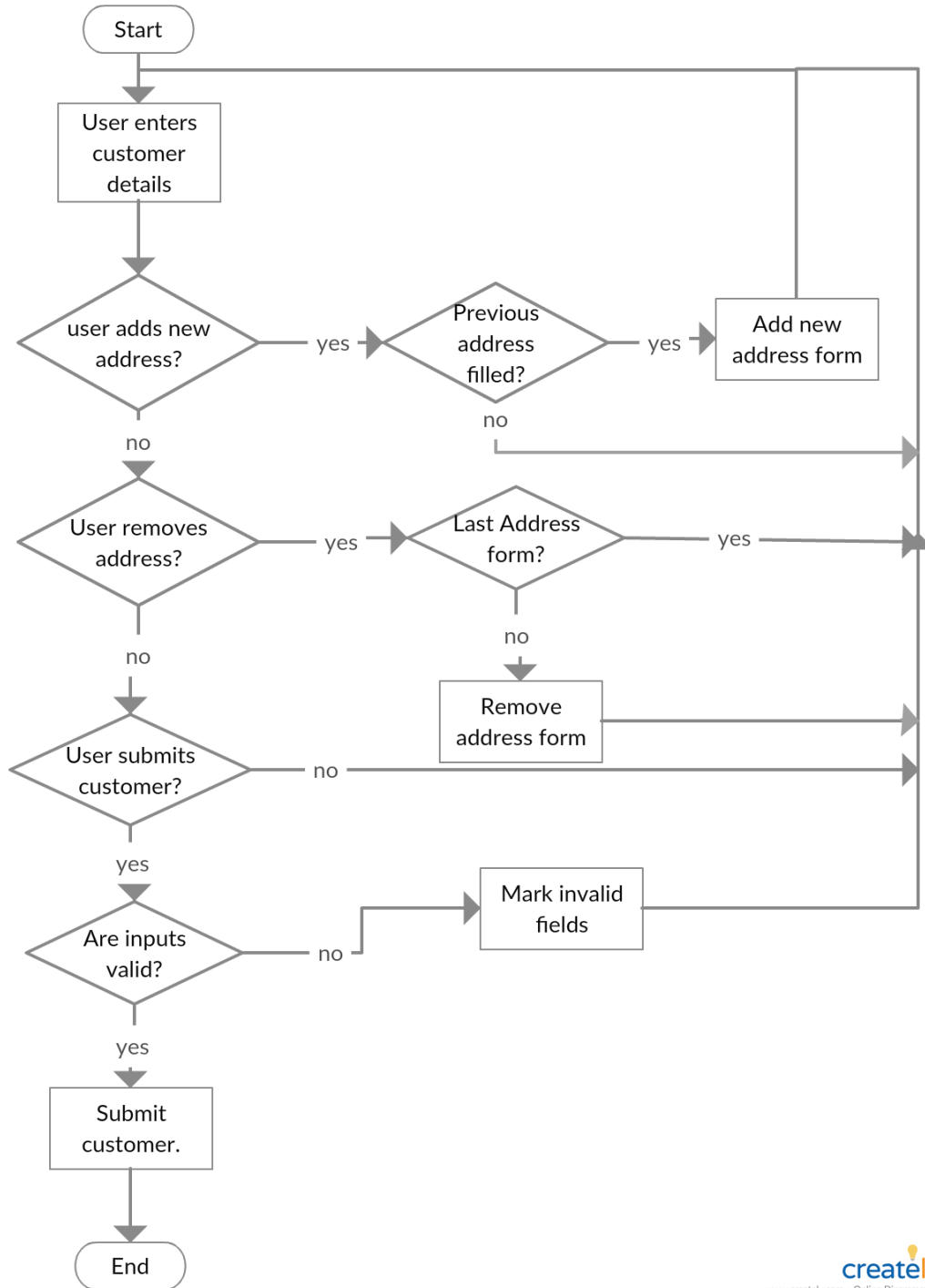


Figure: Entity-Relationship diagram of Rental Inventory Management System

7.2.4: Flowcharts and Algorithms

Add a new Customer



Algorithm:

Step 1: User enters customer details

Step 2: If user selects 'add address' then

 If previous address form is filled then

 Add new address form

 Goto step 1

 End if

End if

Step 3: If user selects 'remove address' then

 If this is not the last address form then

 Remove that address form

 Goto step 1

 End if

End if

Step 4: If user selects 'submit customer' then

 If all form inputs are valid then

 Submit customer details to server

 Goto step 5

 Else

 Mark the invalid inputs

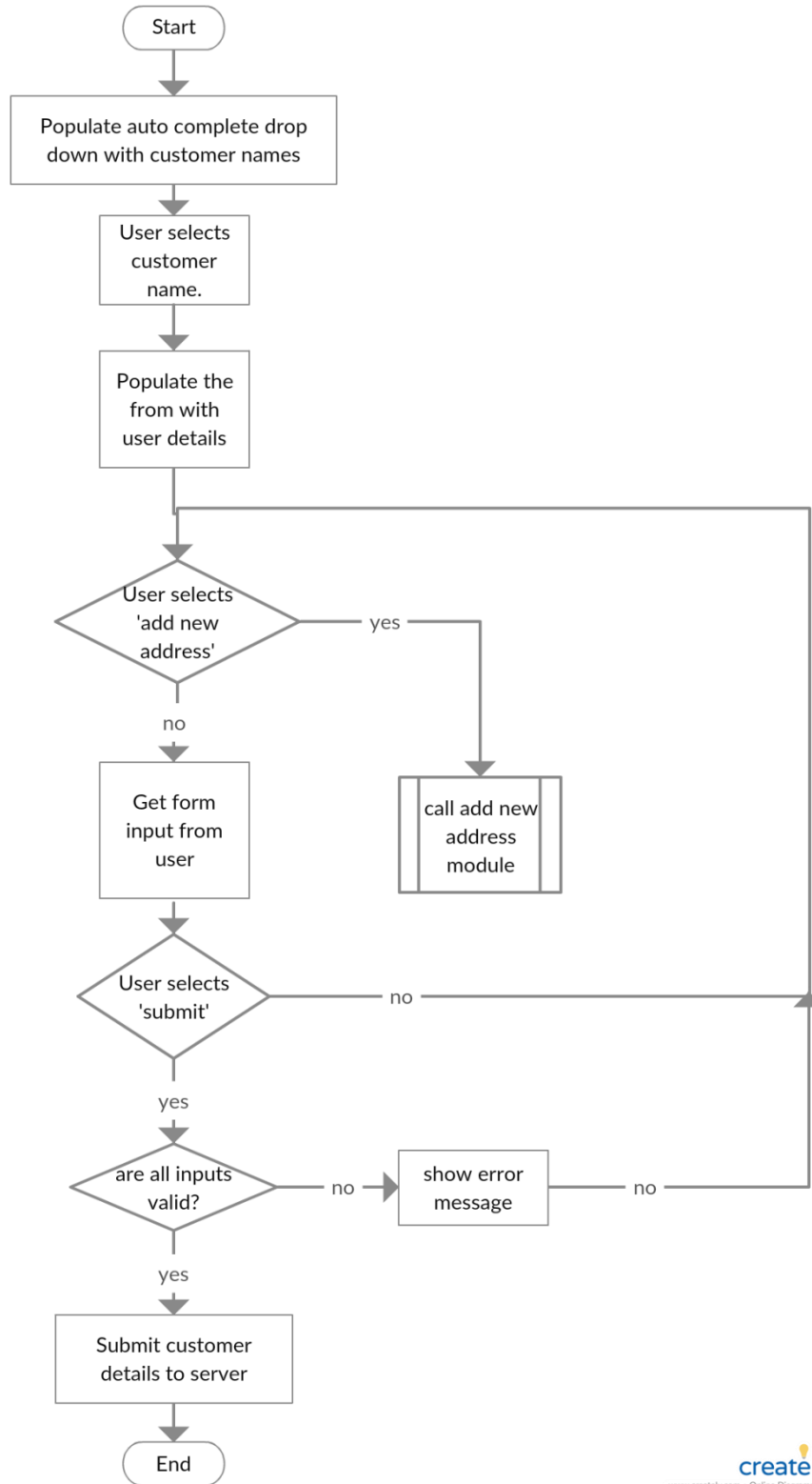
 Goto step 1

 End if

End if

Step 5: End

Manage a Customer Flowchart:



Algorithm:

Step 1: Populate autocomplete drop down with customer names

Step 2: User chooses customer name [by autocomplete searching].

Step 3: Populate the form with previously saved details of the chosen customer

Step 3: If 'add new address' is selected then:

 Call addNewAddress module

 End if

Step 4: User inputs all the form fields [user cannot edit non-editable ones]

Step 5: If user selects 'submit' then:

 If all the inputs are valid then:

 Submit modified customer details to server

 Else

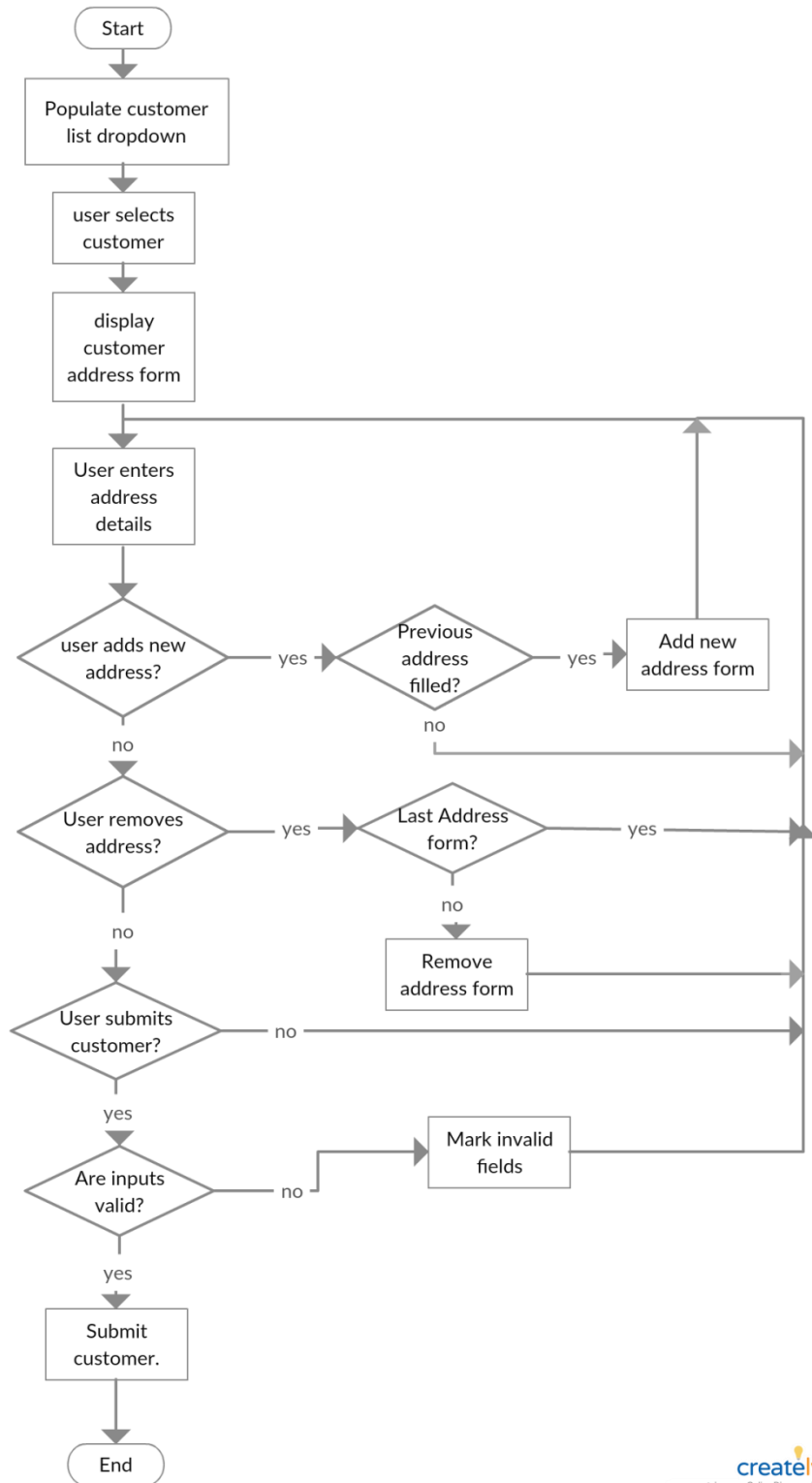
 Show error message

 End if

 End if

Step 6: End

Add new Address to Customer Flowchart:



Algorithm:

Step 1: Populate autocomplete drop down with customer names

Step 2: User chooses customer name [by autocomplete searching].

Step 3: Populate the form with previously saved details of the chosen customer

Step 4: User enters address details

Step 5: If user selects 'add address' then

 If previous address form is filled then

 Add new address form

 Goto step 1

 End if

End if

Step 6: If user selects 'remove address' then

 If this is not the last address form then

 Remove that address form

 Goto step 1

 End if

End if

Step 7: If user selects 'submit customer' then

 If all form inputs are valid then

 Submit customer details to server

 Goto step 8

 Else

 Mark the invalid inputs

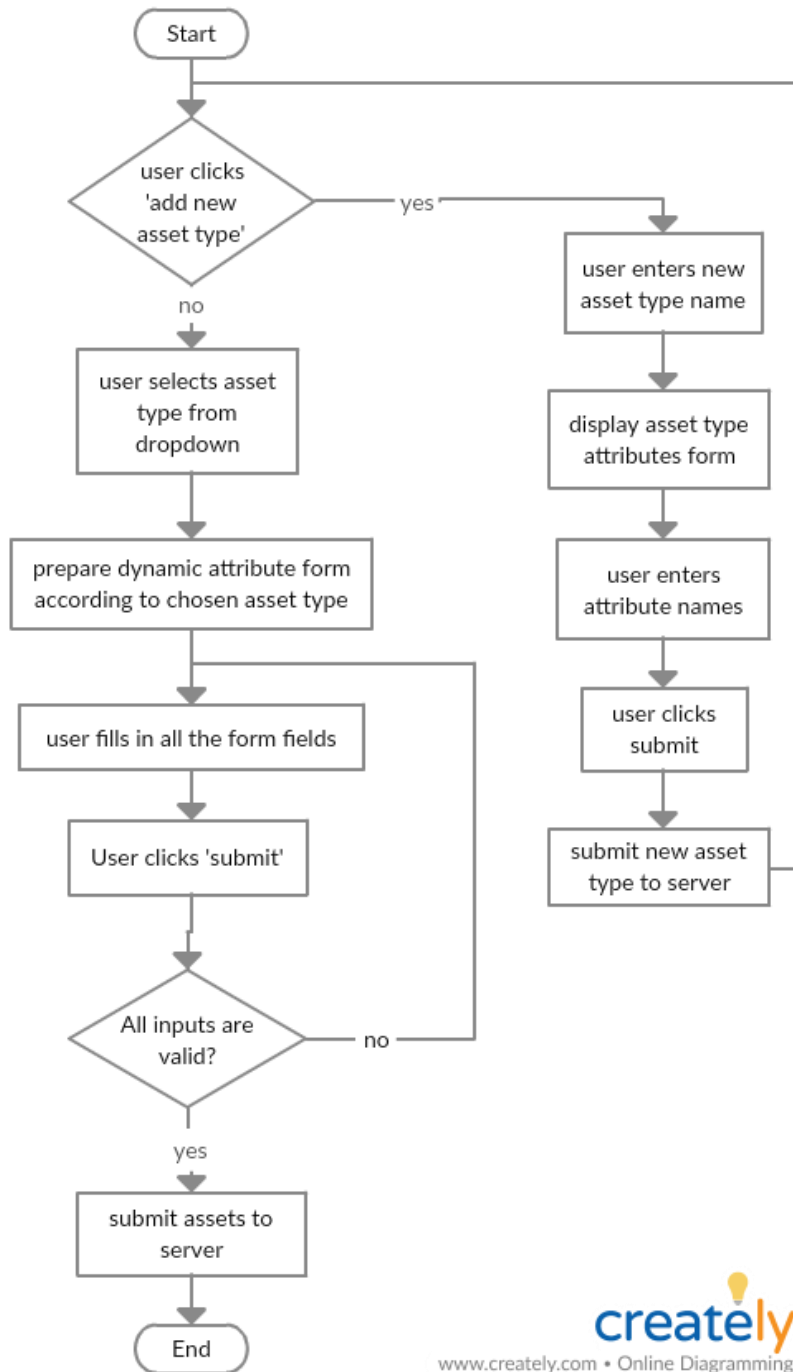
 Goto step 1

 End if

End if

Step 8: End

Add assets to Inventory



Algorithm:

Step 1: if the user clicks 'add new asset type' then:

- a. User enters new asset type name
- b. User enters attribute names of the asset type
- c. User submits asset type
- d. New asset type is sent to server and recorded in database.

End if

Step 2: Populate asset type dropdown with defined asset types.

Step 3: Create the dynamic attributes form based on the asset type selected.

Step 4: User enters all the static and dynamic attribute values of the asset(s).

Step 5: User clicks 'submit'

Step 6: If all inputs are valid then:

Send the asset details along with one or more serial numbers to server.

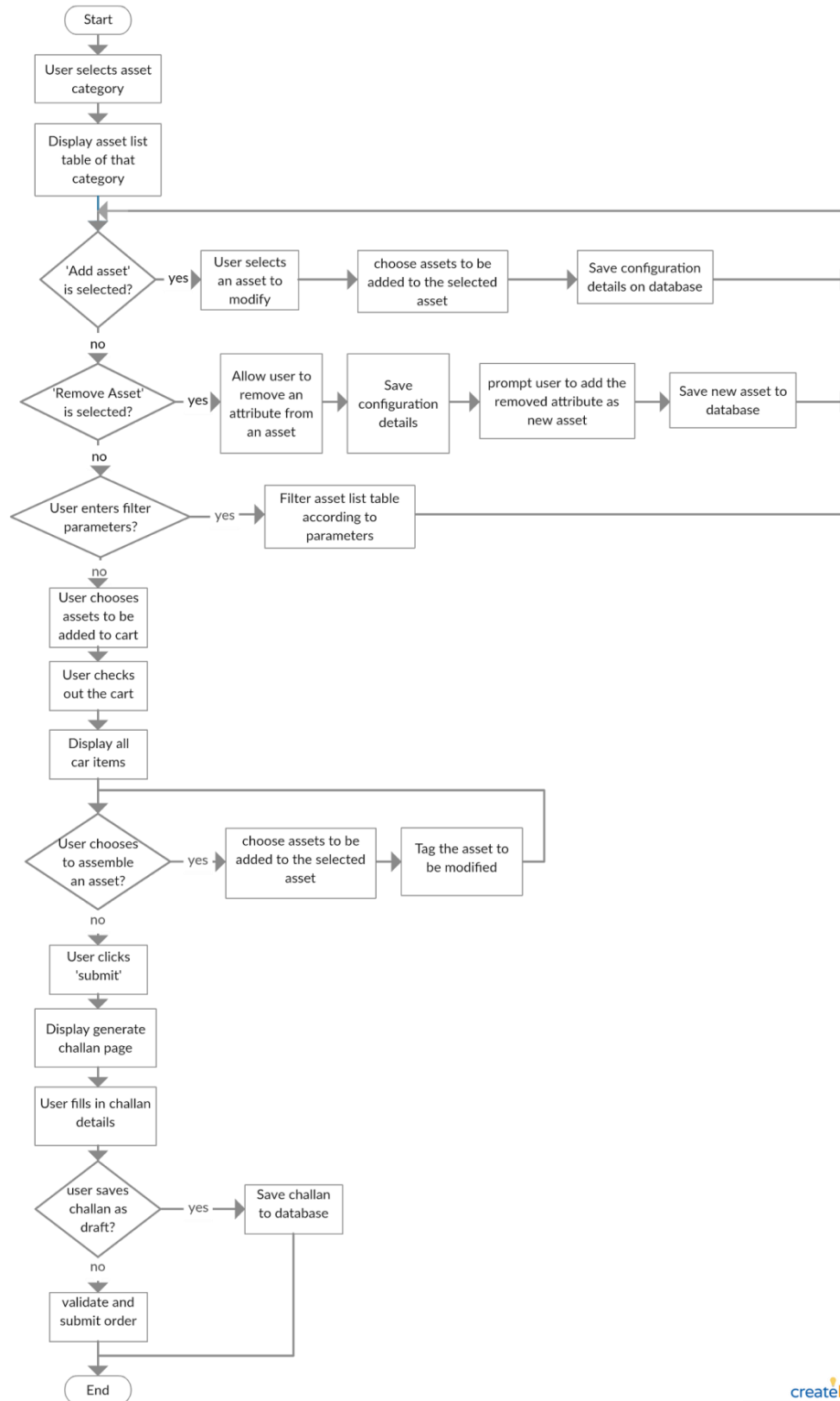
Else if

Show error message

End if

Step 7: End

Order an Asset (Flowchart)



Algorithm

Step 1: User selects an asset category

Step 2: Display asset list table of that category

Step 4: If 'Add asset' is selected then:

- a. User selects an asset to modify
- b. Choose assets to be added to the selected asset
- c. Save configuration details on database

End if

Step 5: If 'Remove Asset' is selected then:

- a. Allow user to remove an attribute from an asset
- b. Save configuration details
- c. Prompt user to add the removed attribute as new asset
- d. Save new asset to database

End if

Step 6: If User enters filter parameters then:

Filter asset list table according to the parameters

End if

Step 7: User chooses assets to be added to cart

Step 8: User 'checks out' the cart

Step 9: Display all car items

Step 10: If User chooses to assemble an asset then:

- a. User chooses assets to be added to the selected asset
- b. Tag the asset to be 'modified'

End if

Step 11: User clicks 'submit'

Step 12: Display generate challan page

Step 13: User fills in challan details

Step 14: If User chooses to assemble an asset then:

Save the challan to database

End if

Step 15: Validate the inputs

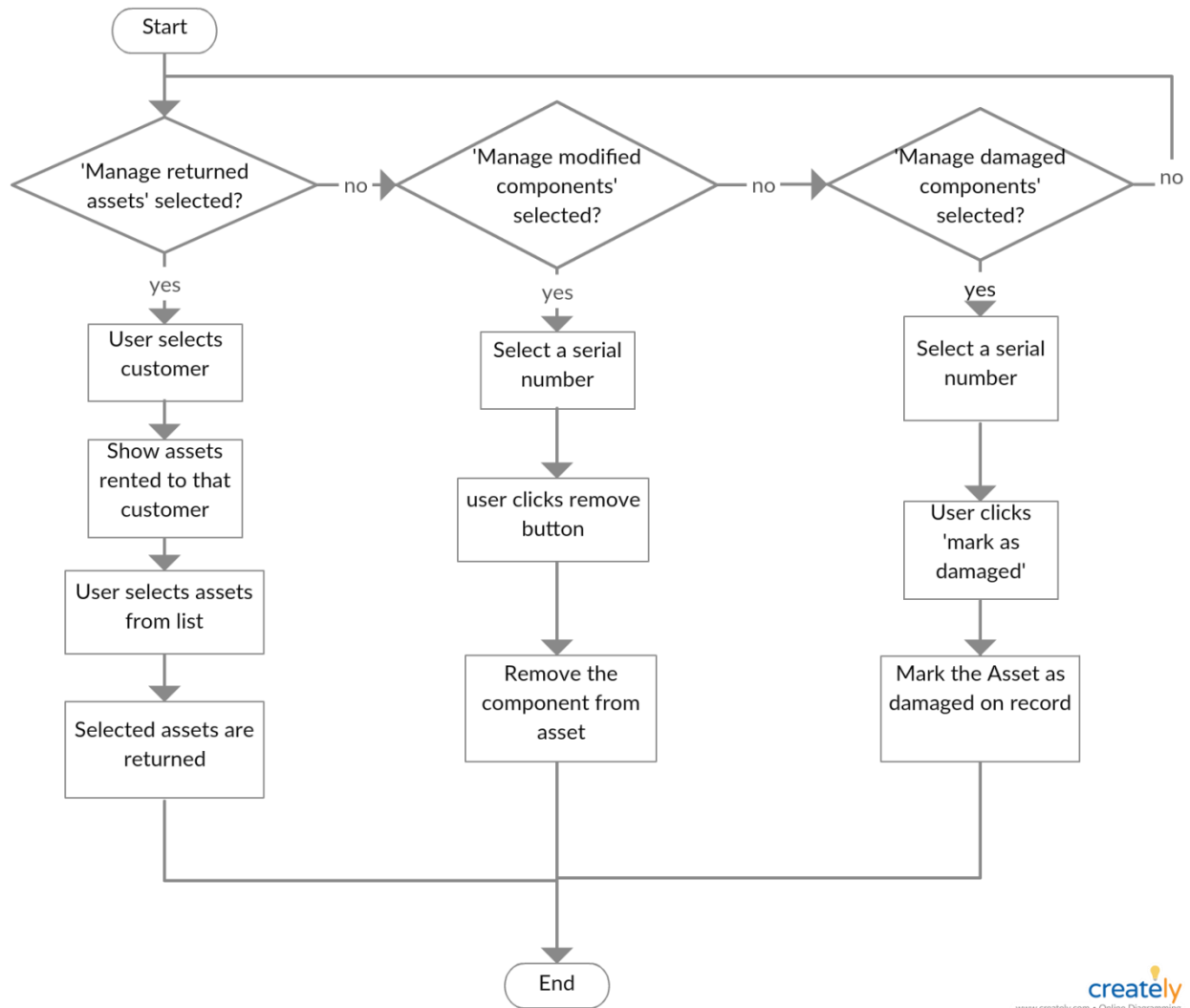
Step 16: Submit the order details to the server

Step 17: Print the challan

Step 18: End

Manage Inventory

Flowchart:

**Algorithm:**

- Step 1: If 'Manage returned assets' selected then:
- User selects customer
 - Show assets rented to that customer

- c. User selects assets from list
- d. Selected assets are returned

End if

Step 2: If 'Manage modified components' selected then:

- a. User selects a serial number
- b. Remove button is clicked
- c. Remove the component from that asset

End if

Step 3: If 'Manage damaged components' selected then:

- d. User selects a serial number
- e. 'Mark as damaged' button is clicked
- f. Mark the Asset as damaged on record

End if

Step 4: End

8: Advantages and Disadvantages of the Proposed System

Advantages:

The following advantages were identified for the proposed system:

- ❑ The front end is developed in React.js which makes it easy and fast to render components, as well as providing various maintainability and scalability features.
- ❑ The back end developed using Node.js provides the distinct advantage of fast data transfer for real time web applications with zero headache regarding sync issues.
- ❑ The system is an upgrade over their existing windows application using visual basic and SQL server which had limited capability and no functionality regarding preparing order challans and creating invoices. In fact there is no provision for creating reports for daily activities.
- ❑ MySQL used to create the database provides a free, flexible, easy to use database with very simple query and table editors.
- ❑ The website is lightweight and highly secure ensuring data integrity for various intra-office functionalities.

Disadvantages:

The following disadvantages were identified for the proposed system:

- ❑ The React.js is just a view layer which means Node.js will run completely asynchronous to React which can be a bit of a problem if the coder is not careful.
- ❑ The learning curve for React.js can be steep and as it is relatively a new technology, useful information on the internet can be hard to come across.
- ❑ There is no provision for a login module as the client has no such setup and hence in version one there is a stark absence of login and registration systems which ensure a reduced profile security. All users are classified as employees and there is no distinction between admin and user modules.
- ❑ It may be necessary to shift to a NoSQL database later on like MongoDB as other complicated features get added to later versions like Payment and Notification features.

9: Future Scope

The proposed system and the document concerns the first version of the project 'Rental Inventory Management System' that we have been required to built during our tenure of internship. The first version of the system is basically a vast upgrade on the existing system of inventory management used by the client company 'Computer Exchange'. The system obviously has a lot of potential and has been built in such a way that there remains a vast room for expansion for future versions. A lot more functionalities has been proposed but has not been integrated into the system as of yet. The project itself has been built as a sort of a template which can be incorporated as just about any inventory management system there is. Here are a few functionalities which can be added into future versions of this system.

- ❖ There was no requirement for a login module by the clients and hence this system has no login and registration system; but as just about any website and also keeping in mind the concerns for data security and to categorise different types of users, a login module may become necessary in the future. Incorporating it into the project will be fairly easy but there must be a requirement for session management which can be brought about by using Redux.
- ❖ The client has mentioned a possibility of adding a payment portal in the future expansions of the system. The payment portal can be especially useful to accept payment in the form of cards or netbanking while preparing orders. Right now the payment has to be completed manually by the customers. Incorporating a payment portal will add greater dynamic, ensuring that the customer need not be involved physically, even in the rental order stage.
- ❖ The client has also mentioned a need for a notification manager in the system which hasn't been included in the first version of the project. The principality of the notification manager is to alert the employees associated with orders and inventory management when a particular customer's rental period is about to end. It may be done in the form of messages on the website or in the form of alert boxes. So to fulfill this a notification system must be incorporated into the existing system.

10: Conclusion

The system has been built keeping in mind the various sensibilities and requirements of our client 'Computer Exchange'. The challenging part of the project was to go ahead and build the project using relatively newer technologies such as React.js and Node.js which made the system design and coding easier, as opposed to building the project using Java. The main aim of the project has always been to provide the client with a much improved inventory management system so as to further automate their daily activities with a zero tolerance for errors. A good inventory management system is essential for smooth functioning of the organisation This system seeks to address those issues at hand by providing a robust inventory management system for intra-office purposes.

11: Bibliography

References:

- <https://stackoverflow.com/>
- <https://www.scribd.com/>
- <https://reactjs.org/docs>
- <https://nodejs.org/en/docs/>
- <https://devcenter.heroku.com/categories/reference>