Group...Class 4C-15

Computer Shop Management System
Software Architecture Document

Version 1.0

Revision History

Date	Version	Description	Author
06/01/2018	1.0	Final Draft.	Nguyễn Thị Hiền

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

1.2 Scope

This document applies to the Computer Shop Management System which will be developed by Group..4C-15.

1.3 Definitions, Acronyms, and Abbreviations

User – a person who use the system, can be customer or employee.

Employee – a person who work for the computer shop.

Accountant – an employee who can save payment of completed order to database.

Guest – a user who is not logged in the system.

1.4 References

None.

1.5 Overview

In the following section, architectural design of the Computer Shop Management System is provided in detail. First, the primary software architecture of the system will be defined. Then, there are further discussion about the goals and constraints that will be imposed upon the quality of the final product, which including but not limited to security, distribution and reuse. In the precedence sections, the key views of the system are demonstrated to depict different aspects of the system. Lastly, criteria concerning with size, performance and quality of the system will be proposed.

2. Architectural Representation

This documents presents the architectural as a series of mandatory views: Use-Case View, Logical View, Deployment View and Data View. These views are presented as Visual Paradigm Community Edition Models , StarUML and use the Unified Modeling Language (UML).

Use-Case View

- **Audience:** all the stakeholders of the system, including the end-users.
- **Area:** describes the set of scenarios and/or use cases that represent significant, central functionality to the system.
- **Related artifacts:** Use-Case Model, Analysis Model, Use-Case-Realization documents.

Logical View

- **Audience:** designers, programmers.
- **Area:** functional requirements: describes the design’s object model.
- **Related artifacts:** Design Model.

Deployment View

- **Audience:** deployment managers, system administrators.

- **Area:** topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects.
- **Related artifacts:** Deployment Model.

Data View

- **Audience:** data specialists, database administrators.
- **Area:** persistence: describes the architecturally significant persistent elements in the data model.
- **Related artifacts:** Data Model.

3. Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

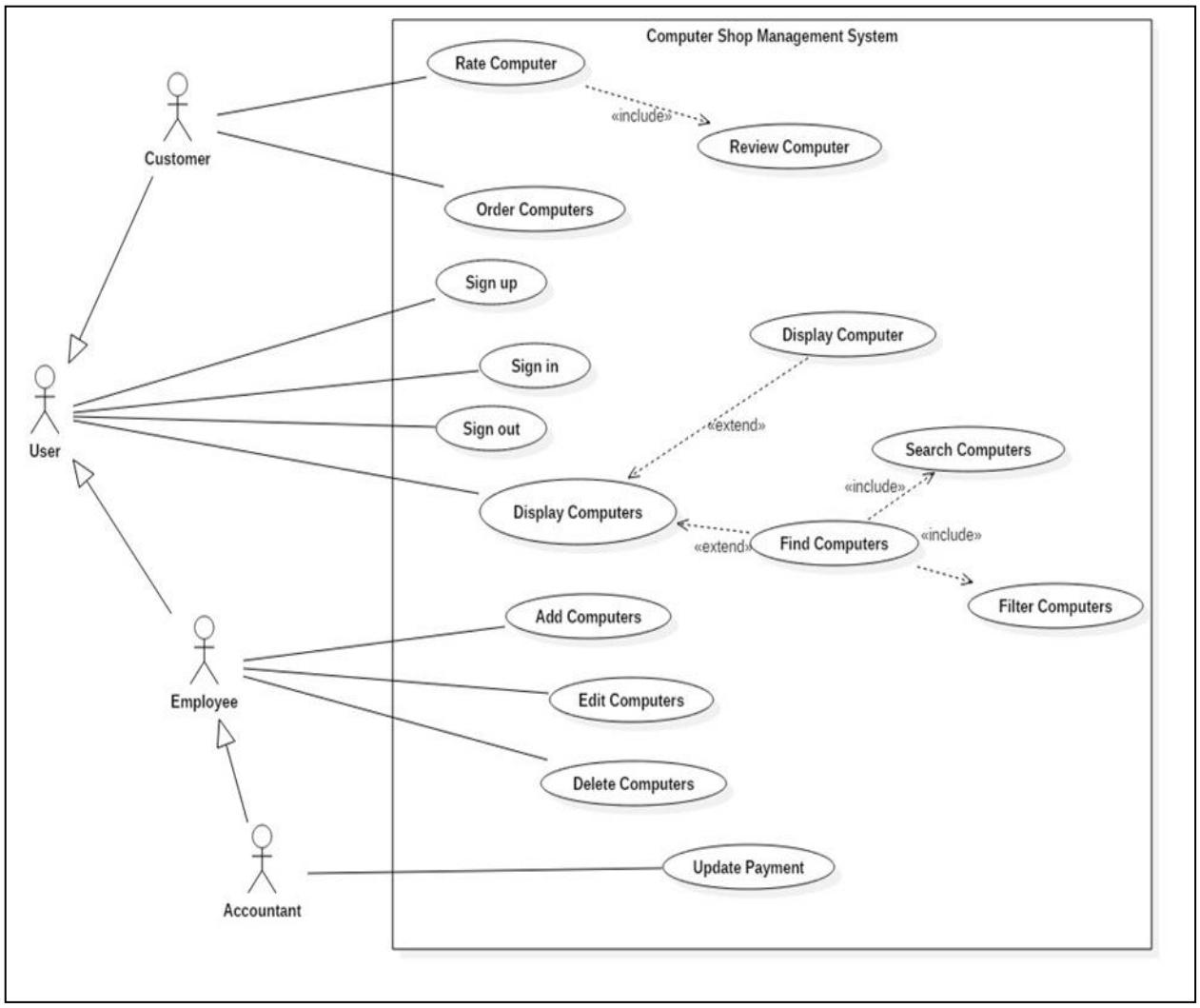
- The Computer Shop Management System must be designed to fulfills all system requirements specified in requirements definition.
- The Computer Shop Management design must be structured to be robust, easy to change if and when functional requirements change.
- The Computer Shop Management System must be designed to allow the re-use of business logic across applications; therefore, the design separate the three components: model, view and controller.
- The separation of the three components: model, view and controller are also necessary to provide a convenient cooperation between different development teams.
- The Computer Shop Management System will run on a dedicated platform with access to a database.
- The Computer Shop Management website provides most of the content display. An interface to this system must be capable of handling large traffic volumes.

4. Use-Case View

A description of the Use-Case View of the system architecture. The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

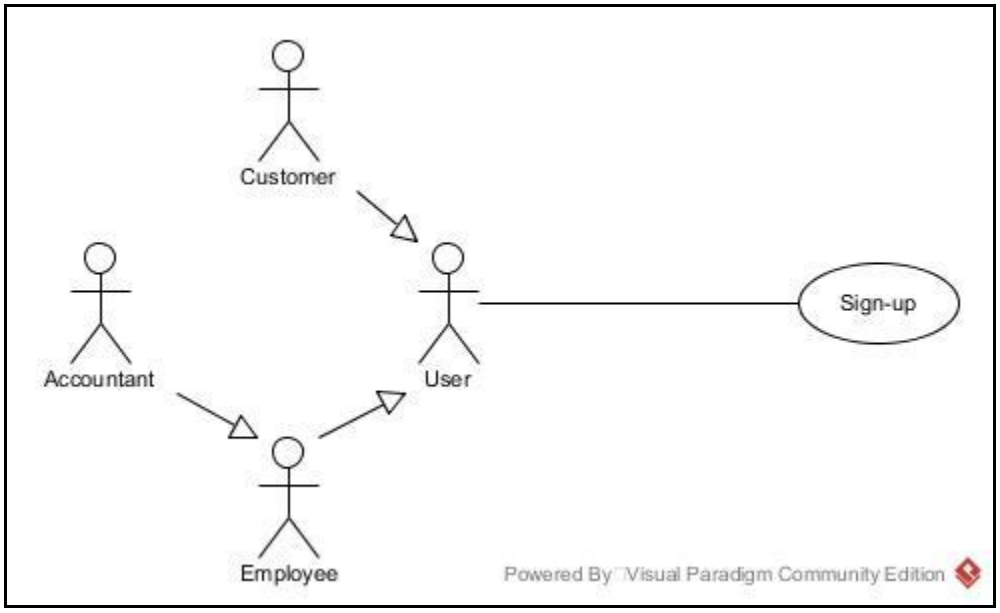
The significant use cases in this system are listed below:

- Sign-up
- Sign-in
- Sign-out
- Display Computers
- Display Computer
- Find Computers
- Order Computers
- Rate Computer
- Review Computer
- Add Computers
- Edit Computers
- Delete Computers
- Update Payment



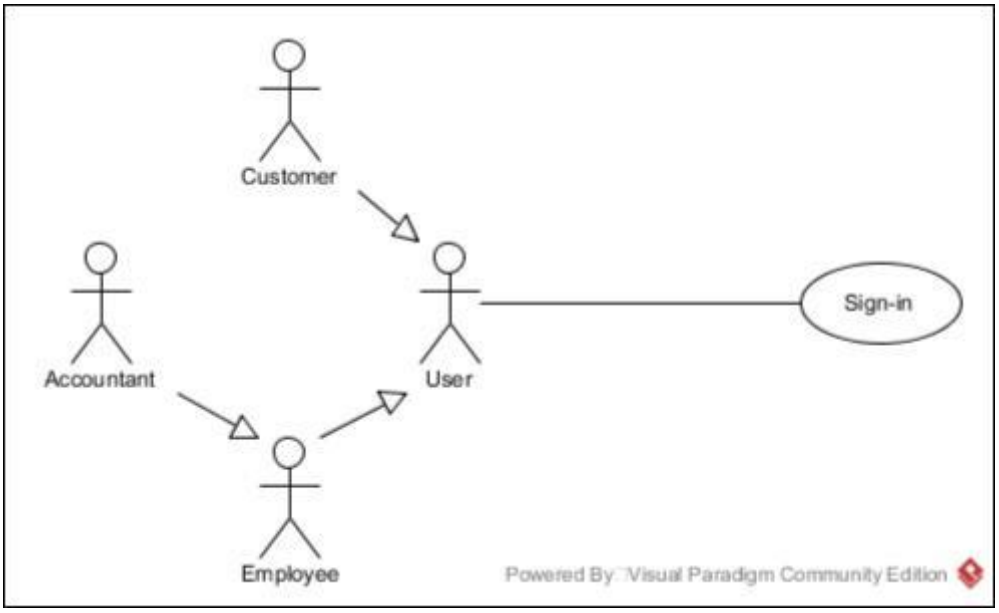
4.1 Use-Case Realizations

Sign-up:



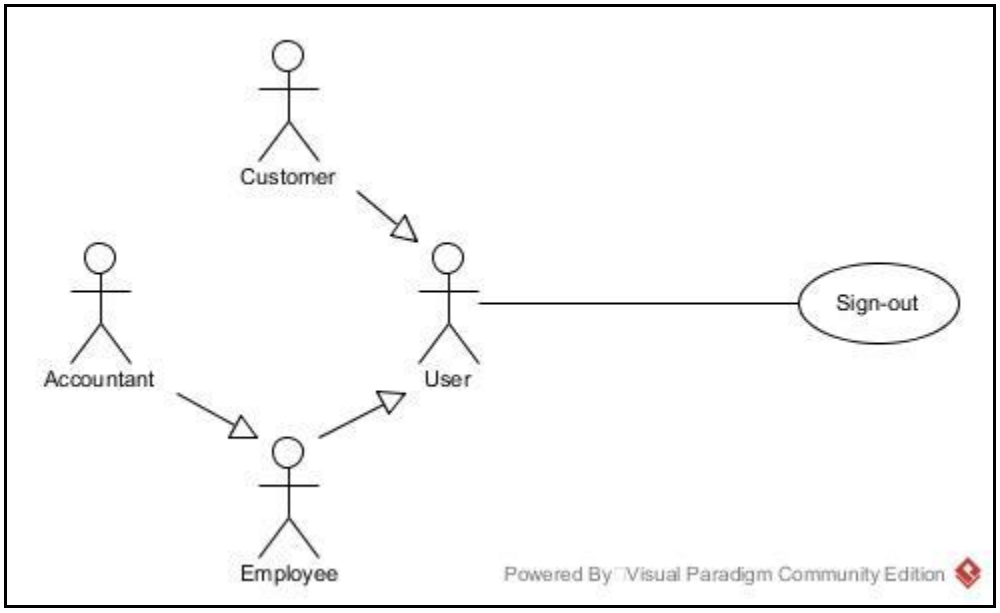
- **Brief Description:** A user creates an account.
- **Specification:** See Use-Case-Realization Specification: Sign-up.

Sign-in:



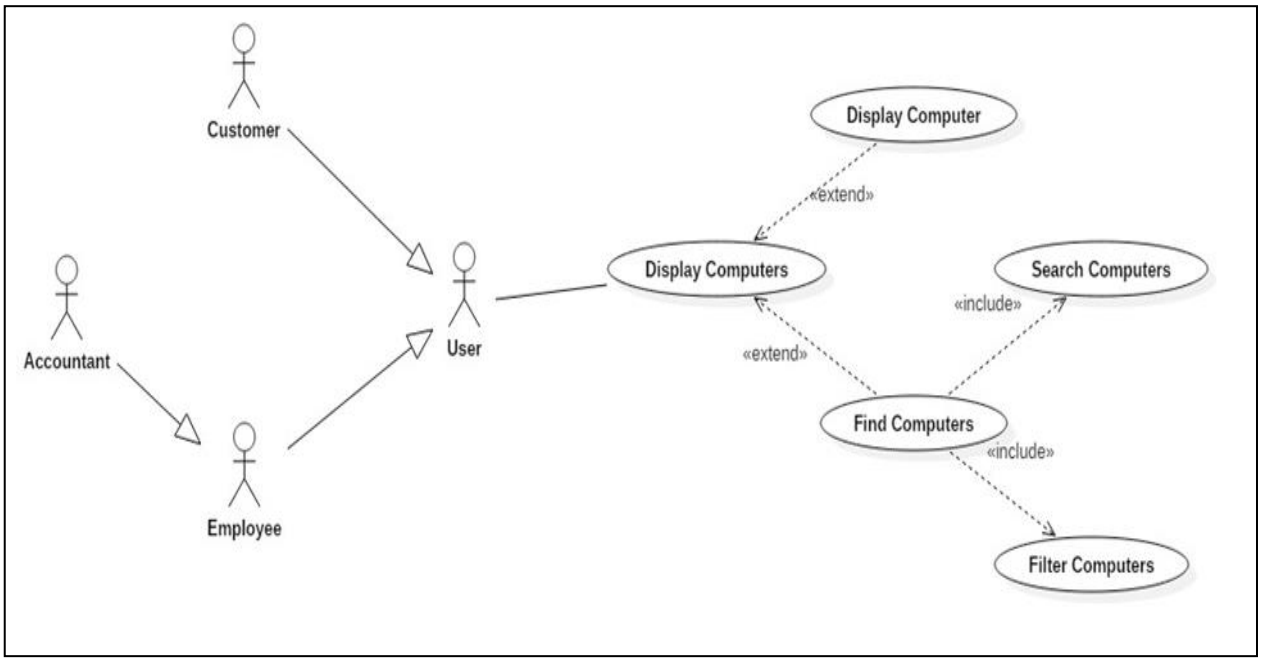
- **Brief Description:** A user logging in to the system.
- **Specification:** See Use-Case-Realization Specification: Sign-up.

Sign-out:



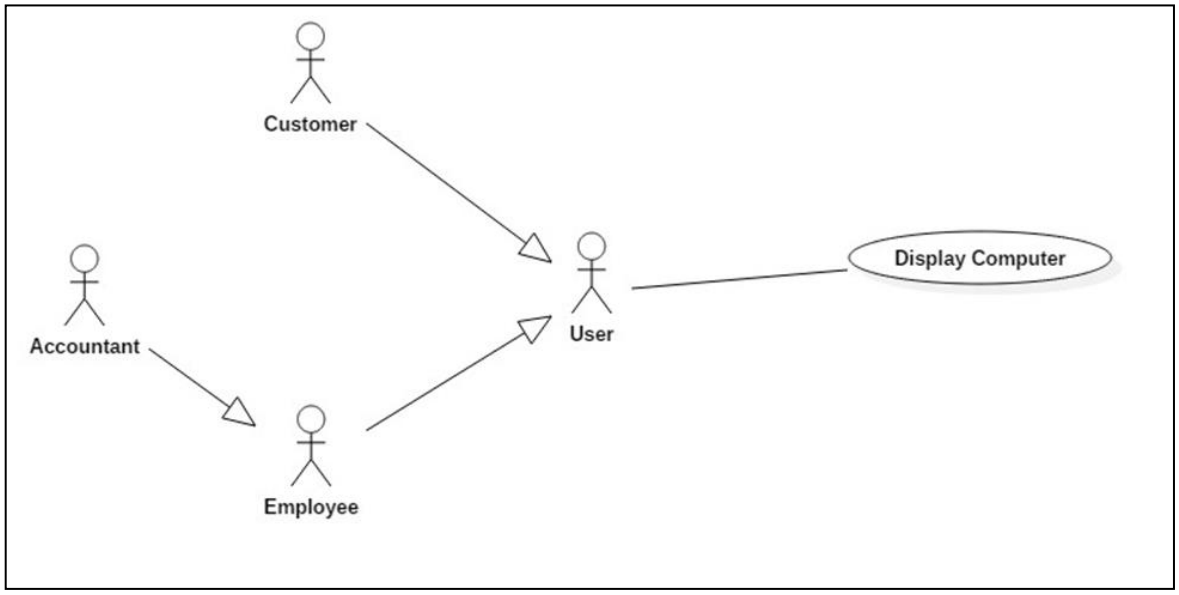
- **Brief Description:** A user logging out the system.
- **Specification:** See Use-Case-Realization Specification: Sign-out.

Display Computers:



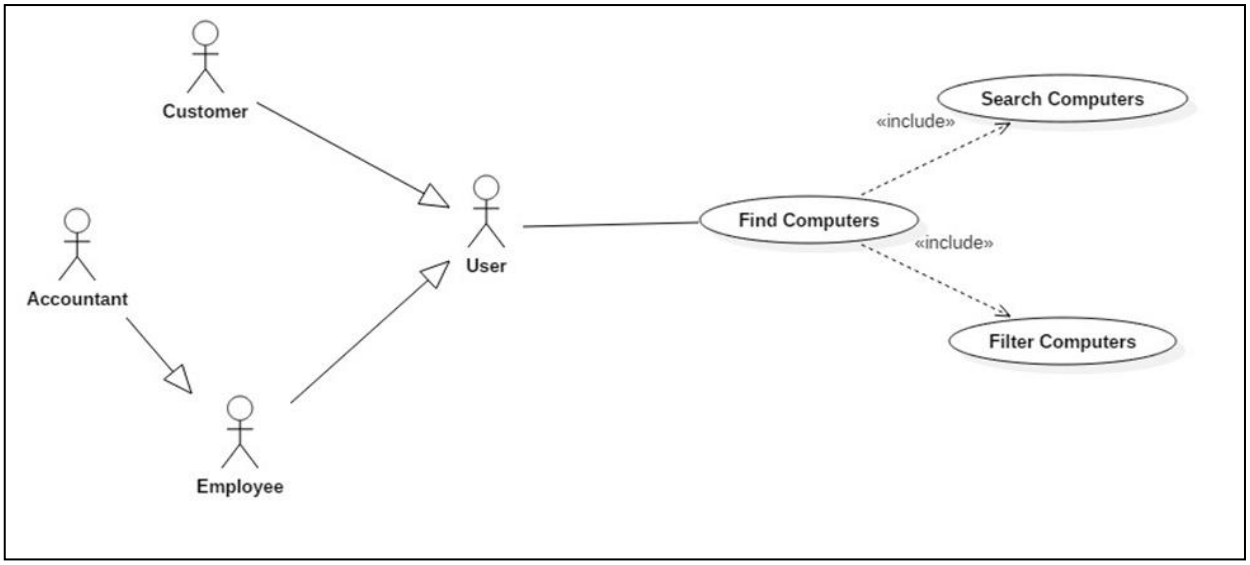
- **Brief Description:** A user displays all available computers of the system.
- **Specification:** See Use-Case-Realization Specification: Display Computers.

Display Computer:



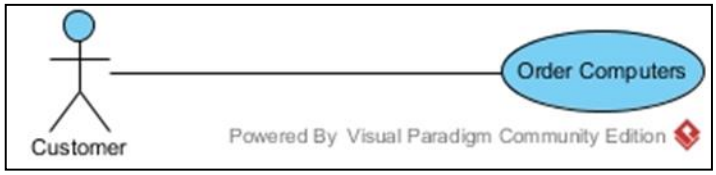
- **Brief Description:** A user displays detailed information of a computer.
- **Specification:** See Use-Case-Realization Specification: Display Computer.

Find Computers:



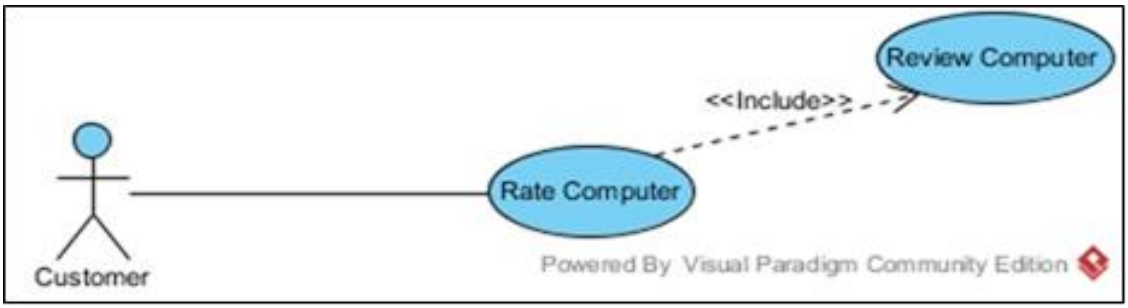
- **Brief Description:** A user displays computers based on search and/or filtering options.
- **Specification:** See Use-Case-Realization Specification: Find Computers.

Order Computers:



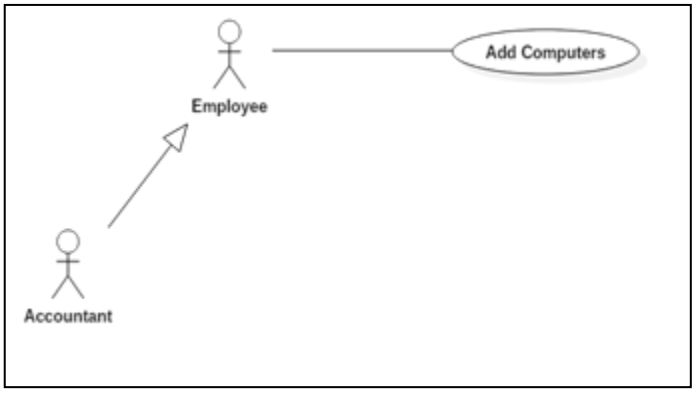
- **Brief Description:** A customer order computer(s) online through the system.
- **Specification:** See Use-Case-Realization Specification: Order Computers.

Rate Computer:



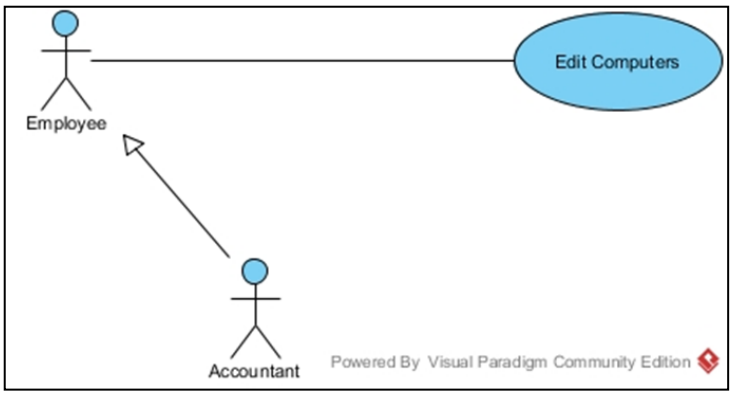
- **Brief Description:** A customer rates a computer.
- **Specification:** See Use-Case-Realization Specification: Rate Computers.

Add Computers:



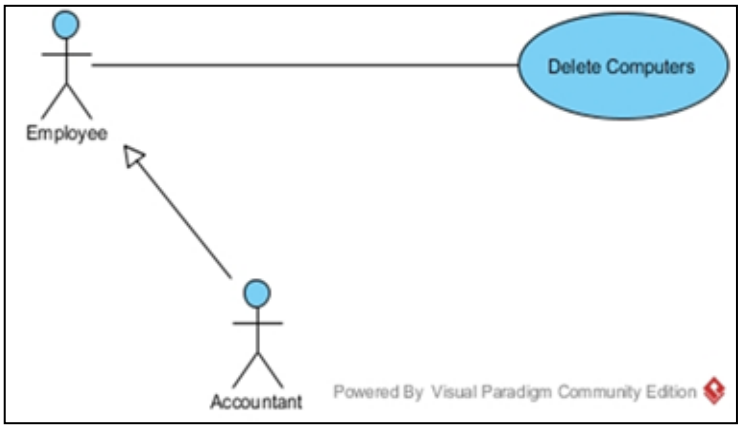
- **Brief Description:** An employee adds new computer(s) to the database.
- **Specification:** See Use-Case-Realization Specification: Add Computers.

Edit Computers:



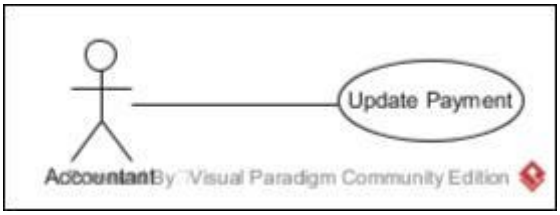
- **Brief Description:** An employee edits existing computer(s) in the database.
- **Specification:** See Use-Case-Realization Specification: Edit Computers.

Delete Computers:



- **Brief Description:** An employee deletes computer(s) from the database.
- **Specification:** See Use-Case-Realization Specification: Delete Computers.

Update Payment:



Brief Description: An accountant update payment information of a completed order.

Specification: See Use-Case-Realization Specification: Update Payment.

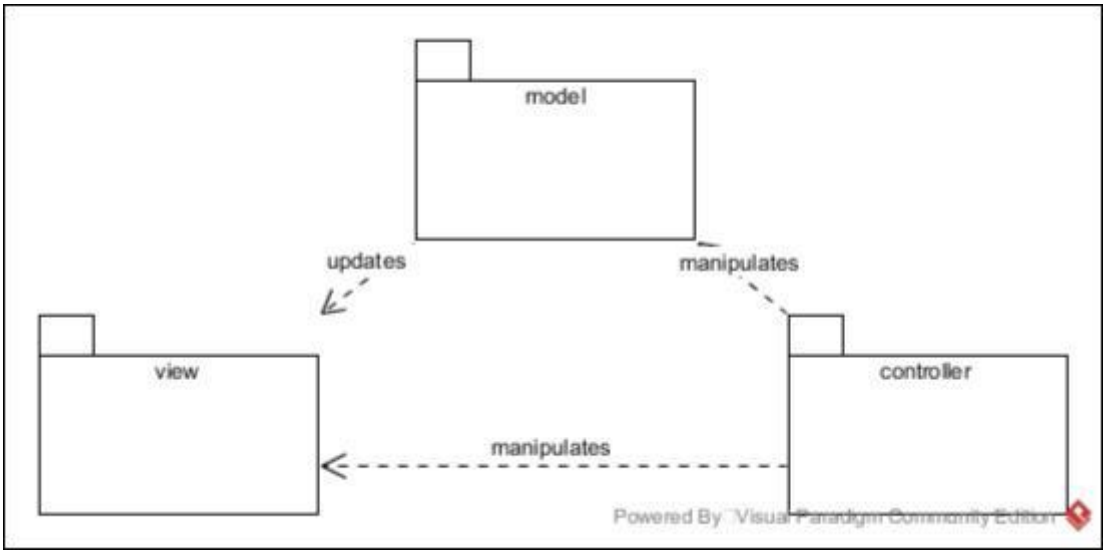
5. Logical View

5.1 Overview

A description of the logical view of the architecture. Describes the overall decomposition of the design model in terms of package hierarchy and layers.

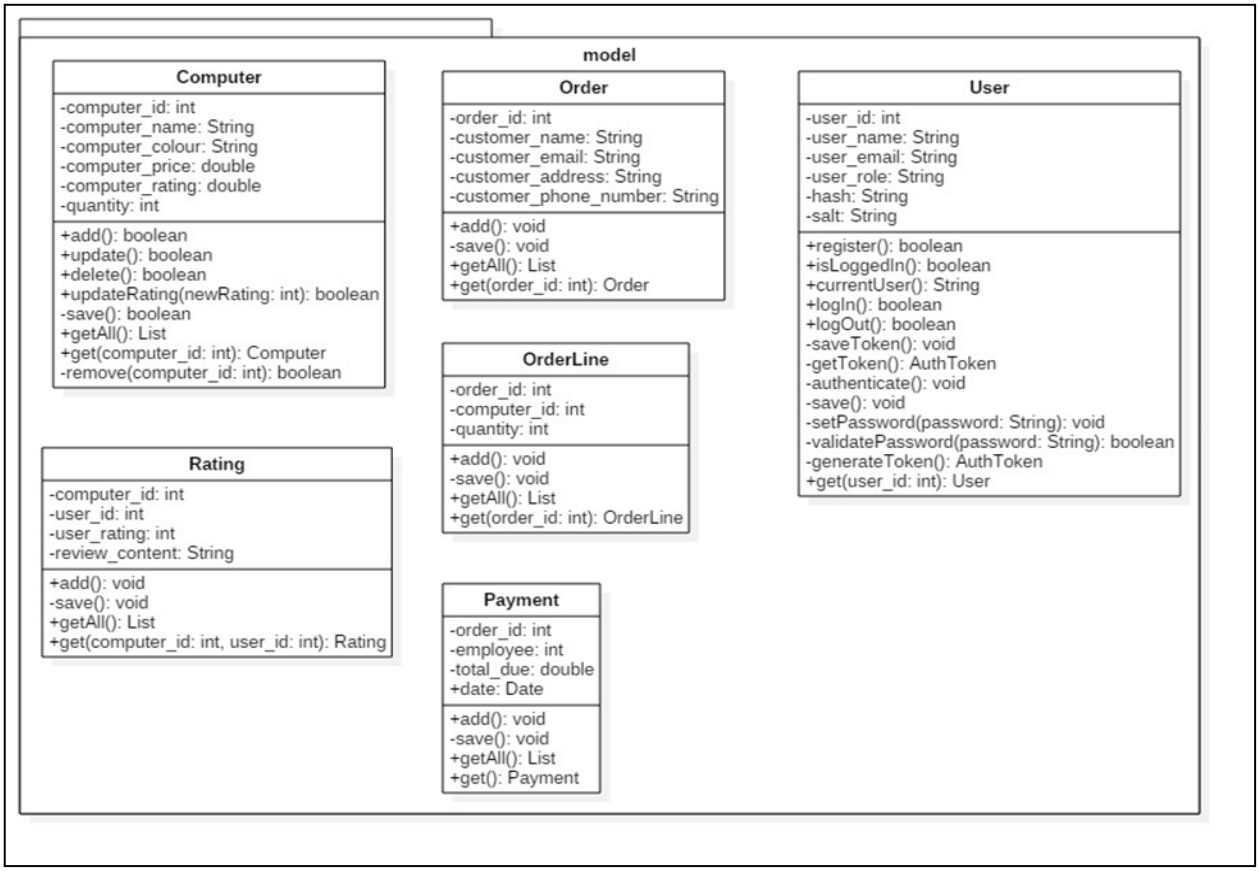
The logical view of the Computer Shop Management System is comprised of 3 significant packages:

- **model:** contains classes that directly manages the data, logic and rules of the Computer Shop Management System and displayed in the view.
- **view:** contains classes that generates output representation of information to the user based on changes in the model.
- **controller:** contains classes that can send commands to the model to update the model's state (e.g., add a new computer); it can also send commands to its associated view to change the view's presentation of the model (e.g., scrolling through computer's reviews).



5.2 Architecturally Significant Design Packages

Package model:



Name	model
Brief Description	Contains classes that directly manages the data, logic and rules of the Computer Shop Management System and displayed in the view.
Classes	Computer, Rating, Order, OrderLine, Payment.

Class Computer:

Name	Computer						
Brief Description	Data model for computer table in database.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
computer_id	int	Private	False	False	N/A	1	N/A
computer_name	String	Private	True	False	50	N/A	N/A
computer_colour	String	Private	True	False	50	N/A	N/A
computer_price	double	Private	True	False	N/A	1	N/A
computer_rating	double	Private	True	True	N/A	1	5
quantity	int	Private	True	True	N/A	0	N/A

Operations				
Header	**Return Type**	**Access**	**Scope**	**Specification**
add()	boolean	Public	Instance	Add the computer this represent to database. Return true if success.

update()	boolean	Public	Instance	Update the computer this represent to datebase. Return true if success.
delete()	boolean	Public	Instance	Delete the computer this represent from database. Return true if success.
updateRating(int)	boolean	Public	Instance	Update current rating of the computer this represent based on new rating and current rating. Return true if success.
save()	boolean	Private	Instance	Save changes from this to database.
getAll()	List	Public	Classifier	Return all computers in database as a List.
get(int)	Book	Public	Classifier	Return a computer in database with specified identifier.
remove(int)	boolean	Private	Classifier	Remove computer in database with specified identifier. Return true if success.

Class Rating:

Name	Rating						
Brief Description	Data model for rating table in database.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
computer _id	int	Private	False	False	N/A	1	N/A
user_id	int	Private	False	False	N/A	1	N/A
user_rating	int	Private	True	False	N/A	1	5
review_content	String	Private	True	True	N/A	N/A	N/A

Operations				
Header	**Return Type**	**Access**	**Scope**	**Specification**
add()	boolean	Public	Instance	Add the rating this represent to database. Return true if success.
save()	boolean	Private	Instance	Save changes from this to database.
getAll()	List	Public	Classifier	Return all ratings in database as a List.
get(int)	Book	Public	Classifier	Return a rating in database with specified identifier.

Class Order:

Name	Order						
Brief Description	Data model for order table in database						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
order_id	int	Private	False	False	N/A	1	N/A
customer_name	String	Private	True	False	50	N/A	N/A
customer_email	String	Private	True	False	250	N/A	N/A
customer_address	String	Private	True	False	250	N/A	N/A
customer_phone_number	String	Private	True	True	15	N/A	N/A

Operations				
Header	**Return Type**	**Access**	**Scope**	**Specification**

add()	boolean	Public	Instance	Add the order this represent to database. Return true if success.
save()	boolean	Private	Instance	Save changes from this to database.
getAll()	List	Public	Classifier	Return all orders in database as a List.
get(int)	Order	Public	Classifier	Return an order in database with specified identifier.

Class OrderLine:

Name	OrderLine						
Brief Description	Data model for order_line table in database						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
order_id	int	Private	False	False	N/A	1	N/A
computer _id	int	Private	False	False	N/A	1	N/A
quantity	int	Private	True	False	N/A	1	N/A
Operations							
Header	**Return Type**	**Access**	**Scope**	**Specification**			
add()	boolean	Public	Instance	Add the order line this represent to database. Return true if success.			
save()	boolean	Private	Instance	Save changes from this to database.			
getAll()	List	Public	Classifier	Return all order lines in database as a List.			
get(int)	Order line	Public	Classifier	Return an order line in database with specified identifier.			

Class Payment:

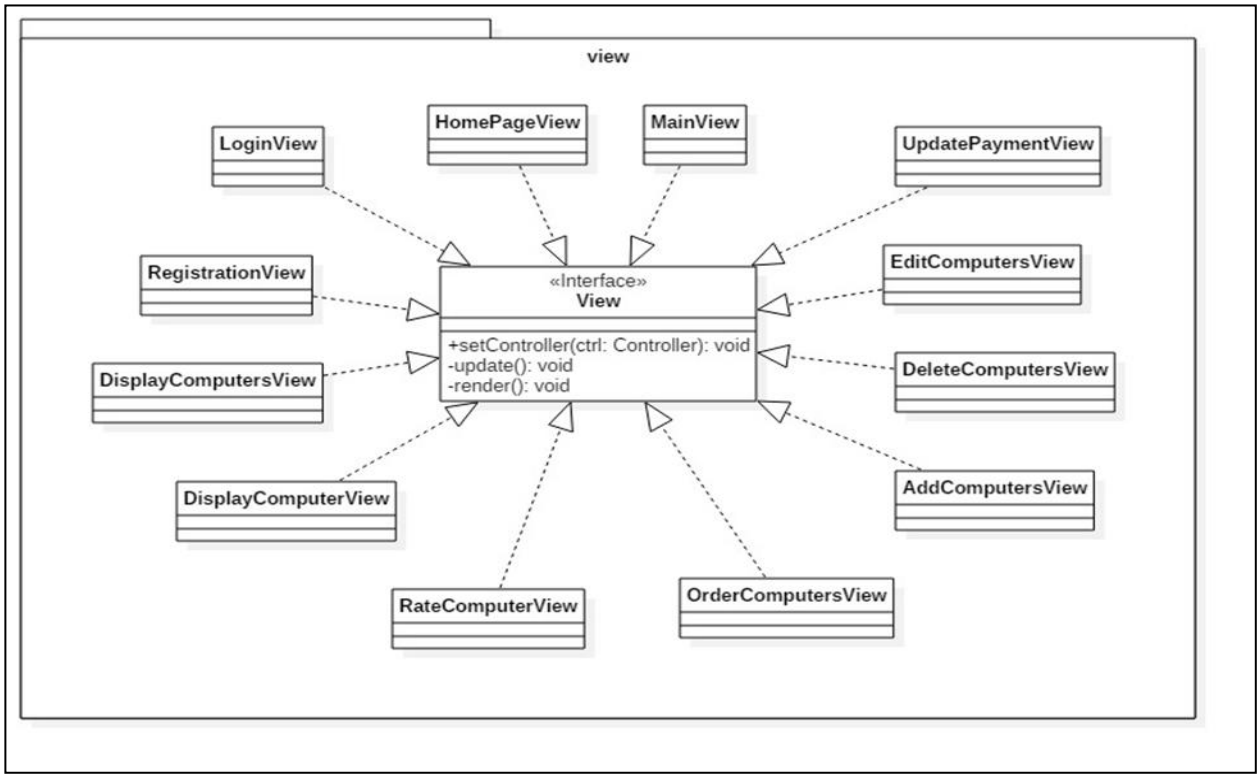
Name	Payment						
Brief Description	Data model for payment table in database						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
order_id	int	Private	False	False	N/A	1	N/A
employee	int	Private	True	False	N/A	1	N/A
total_due	double	Private	True	False	N/A	1	N/A
date	Date	Private	True	False	N/A	N/A	N/A
Operations							
Header	**Return Type**	**Access**	**Scope**	**Specification**			
add()	boolean	Public	Instance	Add the payment this represent to database. Return true if success.			
save()	boolean	Private	Instance	Save changes from this to database.			
getAll()	List	Public	Classifier	Return all payments in database as a List.			
get(int)	Payment	Public	Classifier	Return a payment in database with specified identifier.			

Class User:

Name	User						
Brief Description	Data model for user table in database.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
user_id	int	Private	False	False	N/A	1	N/A
user_name	String	Private	True	False	50	N/A	N/A
user_email	String	Private	True	False	250	N/A	N/A
user_role	String	Private	True	False	15	N/A	N/A
hash	String	Private	True	False	128	N/A	N/A
salt	String	Private	True	False	32	N/A	N/A

Operations				
Header	**Return Type**	**Access**	**Scope**	**Specification**
register()	boolean	Public	Instance	Register the user this represent and save to database. Return true if success.
isLoggedIn()	boolean	Public	Instance	Return true if the current user is logged in the system.
currentUser()	String	Public	Instance	Return the name of the current user.
logIn()	boolean	Public	Instance	Log in the user this represent. Return true if success.
logOut()	boolean	Public	Instance	Log out the user this represent. Return true if success.
saveToken()	void	Private	Instance	Save the authentication token to user local storage.
getToken()	AuthToken	Private	Instance	Get the authentication token from user local storage.
authenticate()	void	Private	Instance	Authenticate the user this represent. Grant the user an authentication token if the user is authenticated.
save()	boolean	Private	Instance	Save changes from this to database.
setPassword(String)	void	Private	Instance	Hash and salt password the user this represent.
validatePassword(String)	boolean	Private	Instance	Validate the password of authenticating user by comparing hash and salt value with existing database hash and salt value in database.
generateToken()	AuthToken	Private	Instance	Generate an authentication token for the authenticated user.
get(int)	User	Public	Classifier	Return an user in database with specified identifier.

Package view:

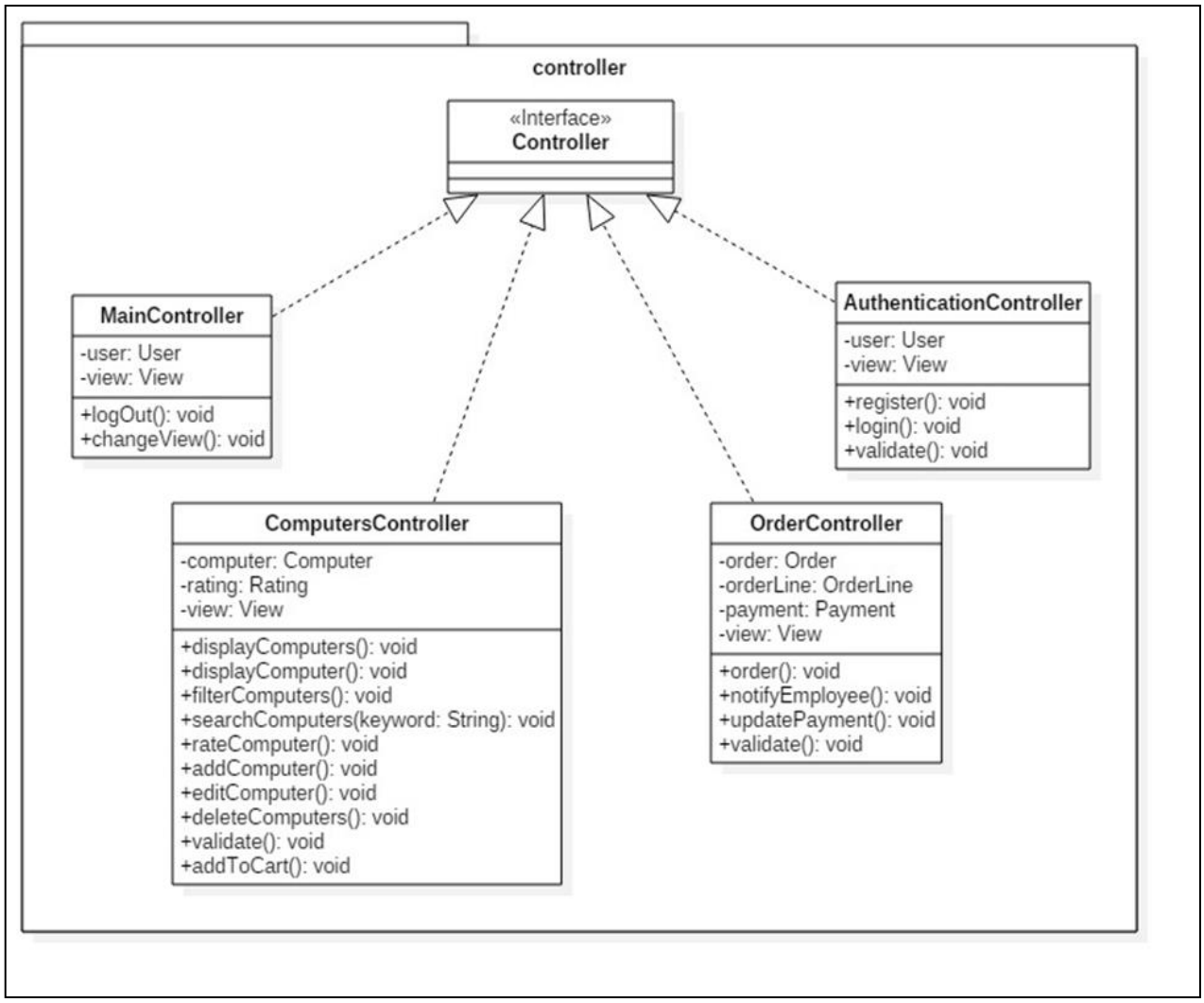


Name	view
Brief Description	Contains classes that generates output representation of information to the user based on changes in the model.
Interfaces	View.
Classes	MainView, HomepageView, RegistrationView, LoginView, DisplayComputersView, DisplayComputerView, RateComputerView, AddComputersView, EditComputersView, DeleteComputersView, UpdatePaymentView.

Interface View:

Name	View			
Brief Description	Represents the visualization of the data that model contains.			
Implementing Classes	MainView, HomepageView, RegistrationView, LoginView, DisplayComputersView, DisplayComputerView, RateComputerView, AddComputersView, EditComputersView, DeleteComputersView, UpdatePaymentView.			
Operations				
Header	**Return Type**	**Access**	**Scope**	**Specification**
setController(Controller)	void	Public	Instance	Map this view with the specified controller.
update()	void	Public	Instance	Update this view based on changes in model.
render()	void	Public	Instance	Render this view.

Package Controller:



Name	controller
Brief Description	Contains classes that directly manages the data, logic and rules of the Computer Shop Management System and displayed in the view.
Interfaces	Controller.
Classes	MainController, ComputersController, OrderController, AuthenticationController.

Interface Controller:

Name	Controller
Brief Description	Controls the data flow into model object and updates the view whenever data changes.
Implementing Classes	MainController, ComputersController, OrderController, AuthenticationController.

Class MainController:

Name	MainController

Brief Description	Controller for the main functionality of the system.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
user	User	Private	true	false	N/A	N/A	N/A
view	View	Private	true	false	N/A	N/A	N/A
Operations							

Header	**Return Type**	**Access**	**Scope**	**Specification**
logOut()	void	Public	Instance	Handling log out request.
changeView()	void	Public	Instance	Handling change view request.

Class BooksController:

Name	ComputersController						
Brief Description	Controller for handling operations related to computers.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
computer	Computer	Private	true	false	N/A	N/A	N/A
rating	Rating	Private	true	false	N/A	N/A	N/A
view	View	Private	true	false	N/A	N/A	N/A
Operations							

Header	**Return Type**	**Access**	**Scope**	**Specification**
displayComputers()	void	Public	Instance	Handling display computers request.
displayComputer()	void	Public	Instance	Handling display computer request.
filterComputers()	void	Public	Instance	Handling filter computers request.
searchComputers(String)	void	Public	Instance	Handling search computers request.
rateComputer()	void	Public	Instance	Handling rate computer request.
addComputer()	void	Public	Instance	Handling add computer request.
editComputer()	void	Public	Instance	Handling edit computer request.
deleteComputers()	void	Public	Instance	Handling delete computers request.
validate()	void	Private	Instance	Validate form inputs.
addToCart()	void	Public	Instance	Handling add to cart request.

Class OrderController:

Name	OrderController						
Brief Description	Controller for handling operations related to order.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
order	Order	Private	true	false	N/A	N/A	N/A
orderLine	OrderLine	Private	true	false	N/A	N/A	N/A
view	View	Private	true	false	N/A	N/A	N/A
Operations							

Header	Return	Access	Scope	Specification

	Type			
order()	void	Public	Instance	Handling customer order computer request.
notifyEmployee()	void	Public	Instance	Notify an employee to deliver ordered items to customer.
updatePayment()	void	Public	Instance	Handling update payment request.
validate()	void	Private	Instance	Validate form inputs.

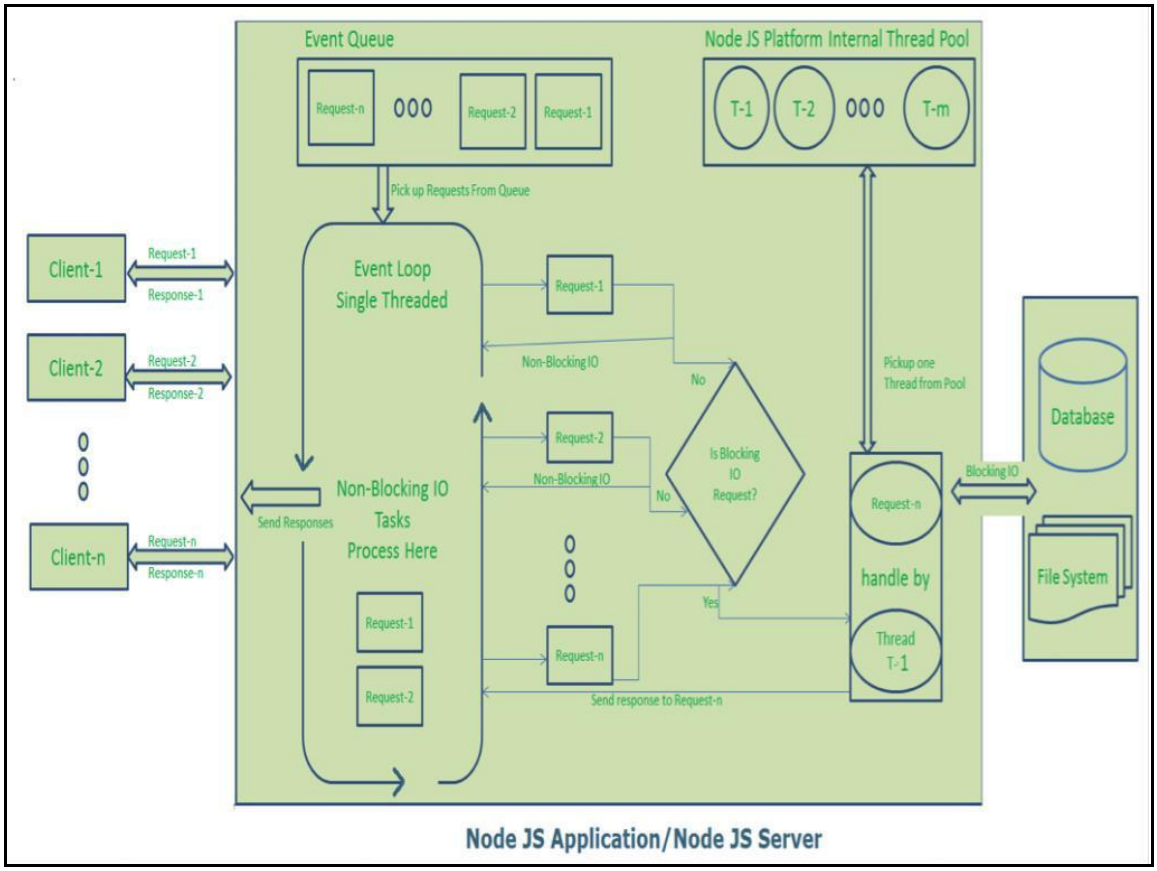
Class AuthenticationController:

Name	AuthenticationController						
Brief Description	Controller for handling operations related to authentication.						
Attributes							
Name	**Type**	**Access**	**Mutable**	**Optional**	**Length**	**Min**	**Max**
user	User	Private	true	false	N/A	N/A	N/A
view	View	Private	true	false	N/A	N/A	N/A
Operations							
Header	**Return Type**	**Access**	**Scope**	**Specification**			
register()	void	Public	Instance	Handling register request.			
logIn()	void	Public	Instance	Handling log in request.			
validate()	void	Private	Instance	Validate form inputs.			

Note: getter and setter methods specification for classes will not be included in this section; nevertheless, appropriate getter and setter for attributes must be generated during implementation based on their domain constraint described above.

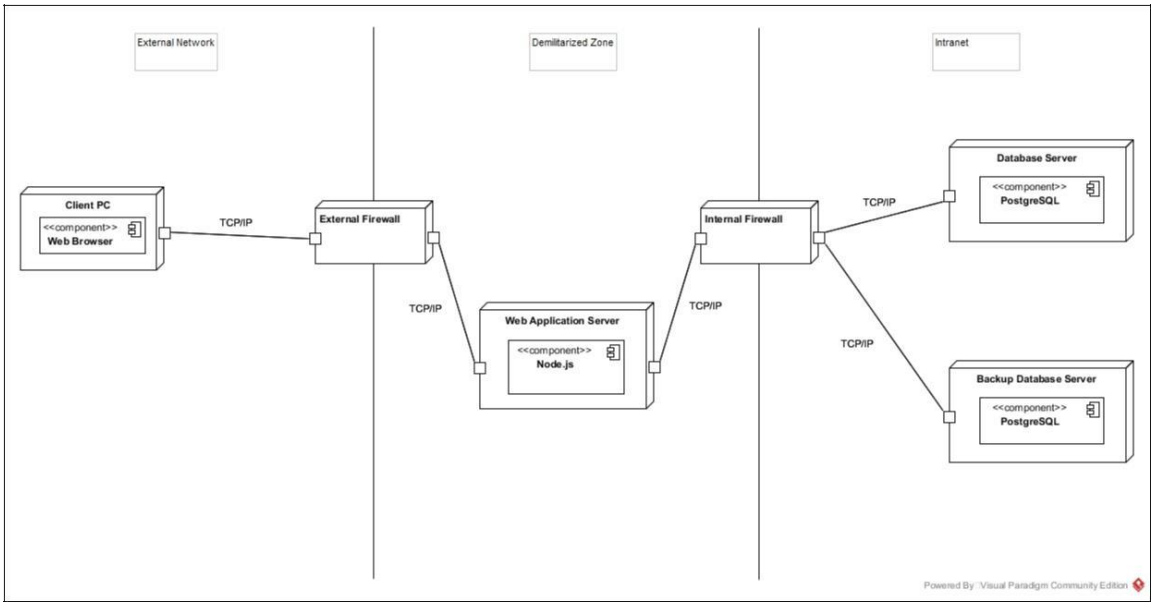
6. Process View

The Computer Shop Management System is designed to be implemented on Node.js server which support single-threaded asynchronous event handling (even loop); therefore, concurrency issues will not be considered in this document.



7. Deployment View

This section describes one or more physical network (hardware) configurations on which the Computer Shop Management System is deployed and run. The system is comprised of these mandatory physical nodes: two firewalls (internal and external), a web server, a database server and a backup database server. The diagram below is the simplicity version of the Computer Shop Management System deployment view.

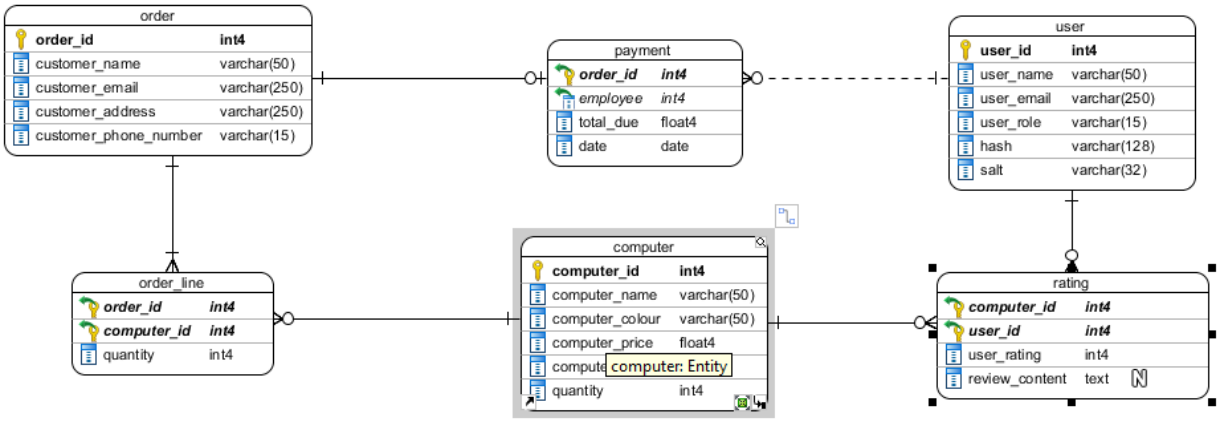


8. Implementation View

The implementation of the system is strictly driven from the design; therefore, the implementation view will not be considered in this document.

9. Data View

A description of the persisten data storage perspective of the Computer Shop Management System



Data Dictionary:

Table	Column	Type	Description	Length	Incl. in PK	Nullable	Unique
computer	computer_id	int4	Unique identifier for each computer.	N/A	True	False	True
	computer _name	varchar	Name of computer.	50	False	False	False
	computer _colour	varchar	Colour of computer.	50	False	False	False
	computer _price	float4	Price of computer.	N/A	False	False	False
	computer _rating	float4	Average rating of computer.	N/A	False	True	False
	quantity	int4	Number of computer s left in stock.	N/A	False	False	False
rating	computer _id	int4	Reference to the computer being rated by user.	N/A	True	False	False
	user_id	int4	Reference to the user that rated the computer.	N/A	True	False	False
	user_rating	int4	Computer rating of user.	N/A	False	False	False
	review_content	text	Content of the review	N/A	False	True	False

order	order_id	int4	Unique identifier for each order.	N/A	True	False	True
	customer_name	varchar	Name of customer.	50	False	False	False

	customer_email	varchar	Email of customer.	250	False	False	False
	customer_address	varchar	Address of customer.	250	False	False	False
	customer_phone_number	varchar	Phone number of customer.	15	False	False	False
order_line	order_id	int4	Reference to an order that this order line belong to.	N/A	True	False	False
	book_id	int4	Reference to the book being ordered.	N/A	True	False	False
payment	order_id	int4	Reference to the completed order of this payment.	N/A	True	False	True
	employee	int4	Reference to user_id of employee who have delivered the ordered items to customer.	N/A	False	False	False
	total_due	float4	Total due of payment	N/A	False	False	False
	date	date	Completed date of payment.	N/A	False	False	False
user	user_id	int4	Unique identifier for each user.	N/A	True	False	True
	user_name	varchar	Name of user.	50	False	False	False
	user_email	varchar	Email of user.	250	False	False	False
	user_role	varchar	Role of user.	15	False	False	False
	hash	varchar	Hash value of user password.	128	False	False	False
	salt	varchar	Salt for user password.	32	False	False	False

10. Size and Performance

The major dimensioning characteristics of the software that impact the architecture and performance constraints:

The system shall support up to 1000 concurrent users against the primary database at any given time, and up to 500 concurrent users against the local servers at any one time.

The system must perform all functions with minimal time delays.

The system must also accurately save all information transactions.

11. Quality

The system architecture supports the quality requirements:

- In order to maintain the highest degree of system integrity, the system is capable of ensuring that all information transitions are saved.
- Databases will be backed up on a daily basis in concern with safety implications.
- The system website is capable of display correctly on different devices web browser of any screen size (i.e. responsive design).
- All system website functions are available through popular web browsers; for instance, Google Chrome, Mozilla Firefox, Opera, Safari, Microsoft Edge, Internet Explorer.