Project Name:

Knowledge Wave

Team Members:

Maitry Patel [215296510]
Komal Patel [215306129]
Karan Pavra [214025704]

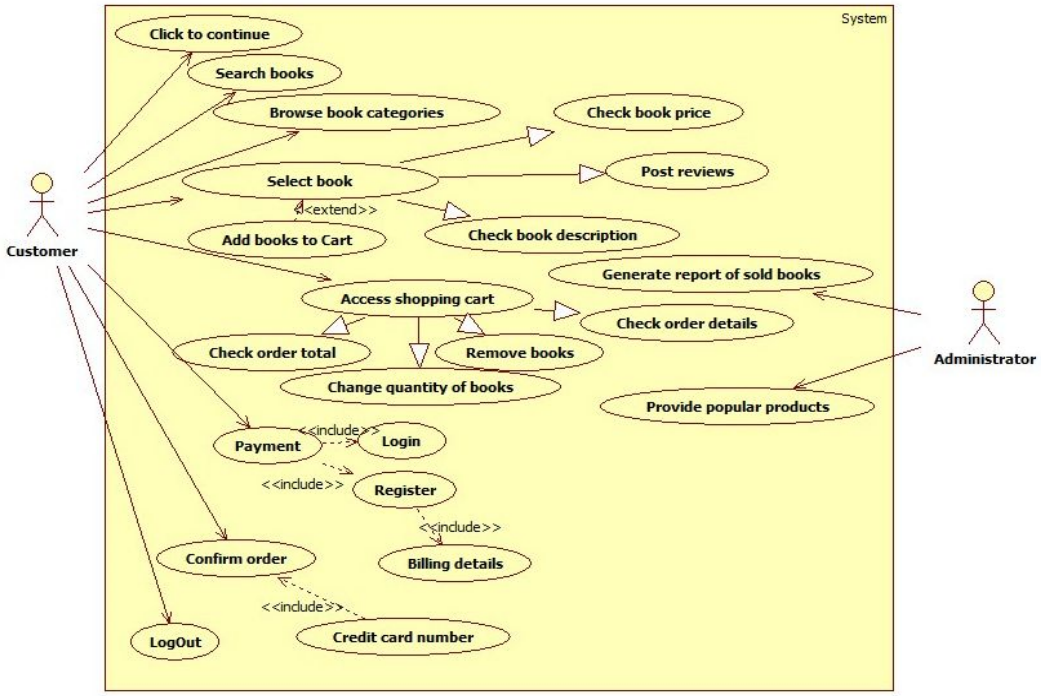
Team Name:

WAVEs

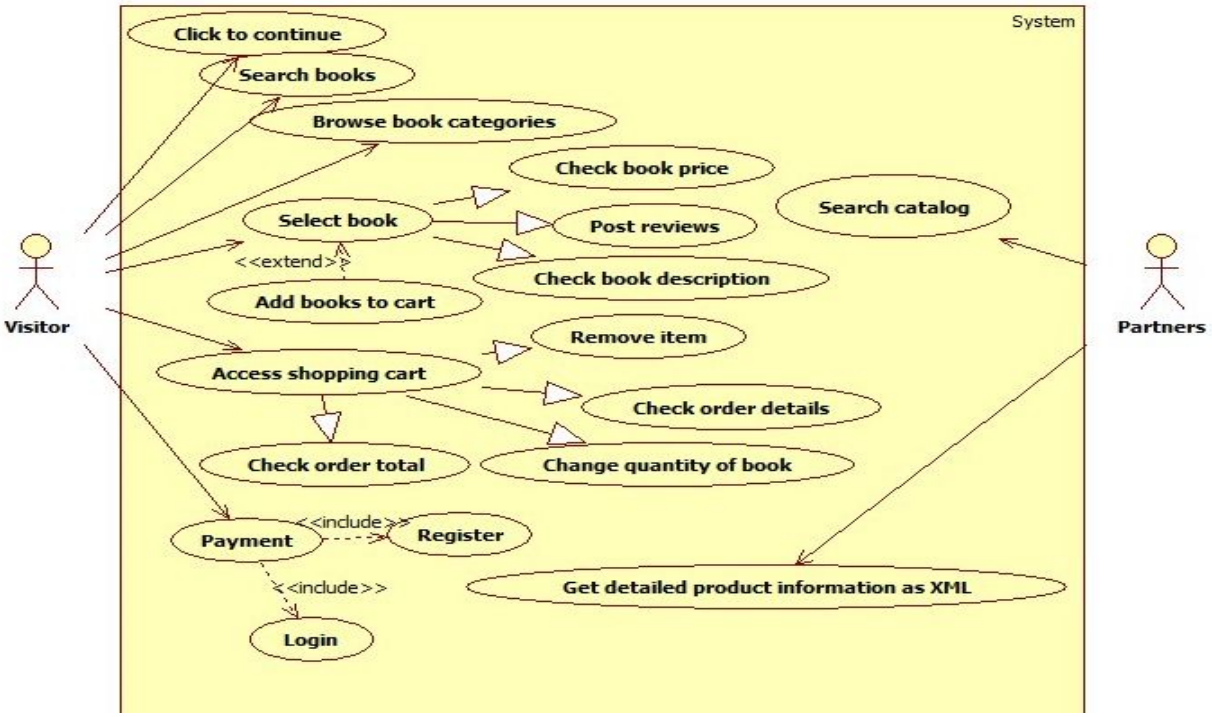
Table of Content:

ARCHITECTURE

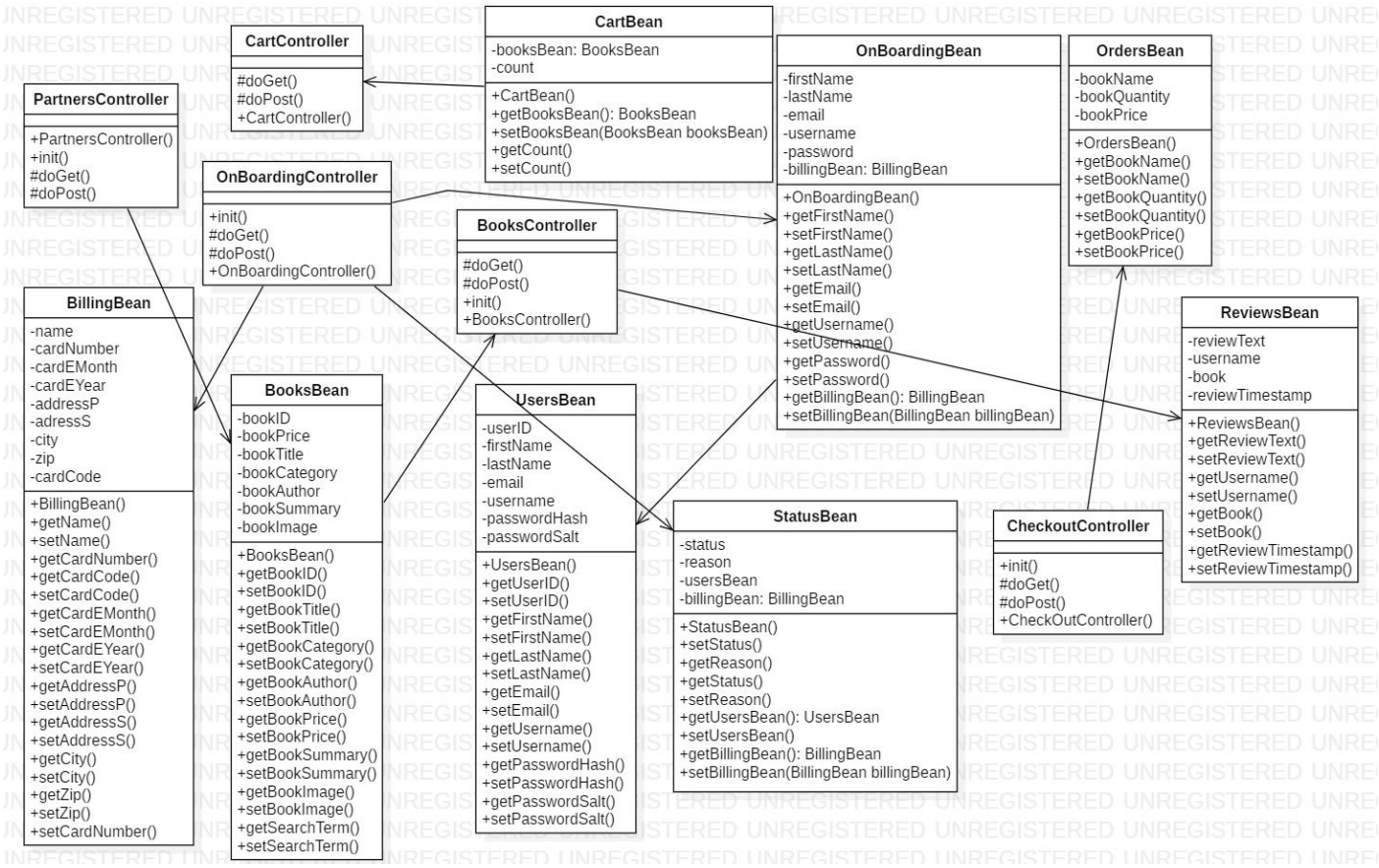
- **UML Use Case Diagram 1:**



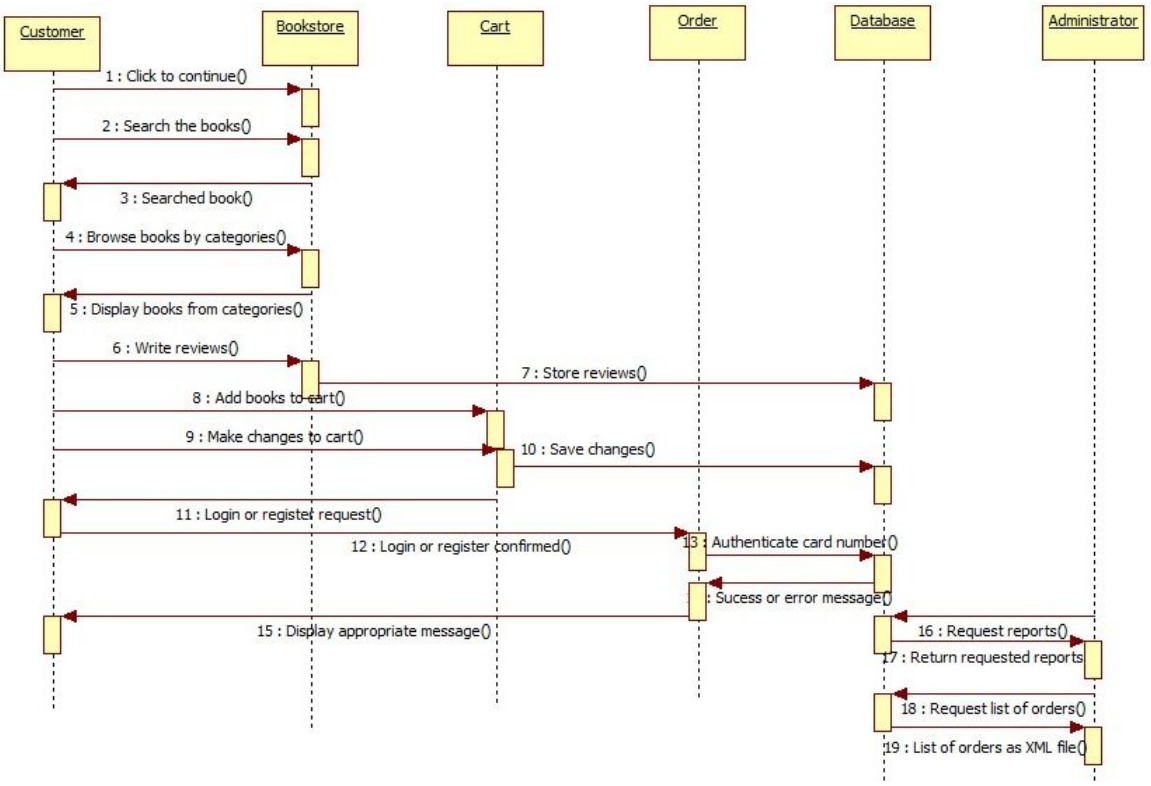
- **UML Use Case Diagram 2:**



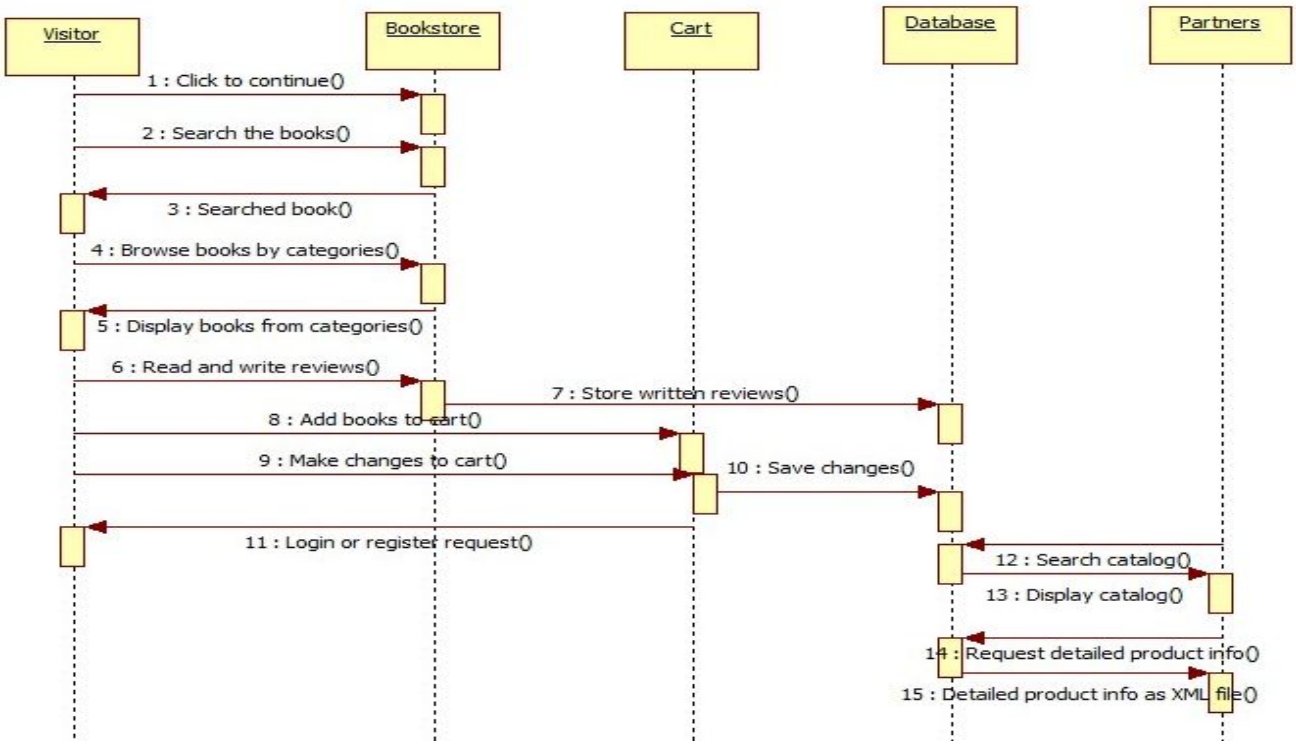
- **Class Diagram:**



- **Sequence Diagram 1:**



- **Sequence Diagram 2:**



DESIGN DECISION

- **ERROR HANDLING**
  All the errors related to database are handling in DAO classes. For rest of the client side errors are handling in js. There are an errors which is related to server side which we throw those errors by exception handling.

- **EVENT DRIVEN DESIGN**
  As per class diagram, all model class design using event driven design for this e-commerce bookstore system.

- **SERVICE ASPECT**
  The services can be grouped as follows:Servlet, Json, Jersey are used as client interfaces. To effectively handle data and communication with database, some other components has to be used uniformly.

- **WEB PAGE DESIGN**
  a) Ajax Implementation : It is also known as a single app design implementation. It improves client experience. The client receives immediate feedback on their actions and it reduces the amount of refreshing page to obtain output of the action performed. When the client adds an item to the cart, the cart number is automatically updated. Also, If client want to give an review for the book they can write without redirecting the page. When the client tries to increase or decrease the quantity of the items added to the cart, the quantity gets updated instantly and the total price changes accordingly.
  b) HTML,jspx : The browser read text from file and translate that text into graphical visual format. Javascript is used to create dynamic pages and provide interaction between client and the application. HTML and javascript are used as designing tool for Intro page, Bookstore main page, Cart page, Payment page, Login/Registration page.

- **DESIGN FOR HIGH CONCURRENCY**
  We are using most popular and open source database MySQL. With the help of that, we can add data throw insert like query and for changes in data we use delete, update, etc. For fetching data from the dataset select is useful.

- **MVC :**
  Model View Controller design pattern divides an application into three different components which are listed below. It allows multiple views and multiple developer can work on these components simultaneously.
  Model-BillingBean.java,BooksBean.java,CartBean.java,OnBoardingBean.java, StatusBean.java, UsersBean.java
  View - Books.jspx, Cart.jspx, Checkout.jspx, Error.jspx, OnBoarding.jspx, Partners.jspx
  Controller-BooksController.java,CartController.java,CheckoutController.java, OnBoardingController.java

- **SPECIAL REQUIREMENT**

  a) For payment page, " hard code that every 3rd request is denied on your website".

  Hard coding is performed on the order checkout page where number of requests are counted in session scope.

  b) Utils.java - password security. To protect password in database, we have used salting algorithm.

  Salt is generated which is unique for each password.

  Salt and plain text is mismatched and it could be referred as hashed password. Using SHA1, salt is matched with hashed password stored in database.

  c) Why we used GSON and JSON both?

  Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object.

IMPLEMENTATION

- **DATA ACCESS COMPONENT**

  How there is connection between application business logic and database

  DAO classes: BooksDAO.java, BooksDAOImpl.java, OrdersDAO.java, OrdersDAOImpl.java, UserDAO.java, UsersDAOImpl.java

- **PRODUCT CATALOG COMPONENT/SERVICE**

  Link: http://localhost:8080/BookstoreWeb/partners

- **ORDER PROCESS COMPONENT/SERVICE**

  It is offered both as a browser action and as a service. Also,it is offered as a service for external partners.

  Link: http://localhost:8080/BookstoreWeb/api/books

- **SESSION CONTROLLER**

  For the Session Controller, we have used Tomcat. This session controller only load when we need the session information.

- **BOOK STORE MAIN PAGE**

  Link: http://localhost:8080/BookstoreWeb/books

  Browse Book Categories : Drop down menu after search input text field

  Select a book and see information : Click on the book photo

  Add a review : For adding review for specific book person need to go click on that book and write review by clicking on review link. Also customer/visitor can see all the reviews posted by previous customer.

  Search the store : You can search any book available in the store.

Add item to shopping cart : First click on the book which you like to purchase. After that you can find “Add to cart” button to add your book in shopping list.
Shopping Cart Button : It is located at bottom right corner of the page. This button also indicate total number of book added to cart.

- **SHOPPING CART PAGE**
  It shows all the book which client added for purchase.
  Increase/decrease quantity, bill updation : Person can Increase/decrease the number of the book they wanted or delete the book if they do no want. There is a validation for book count cannot decrease to zero.
  When the item is deleted from the cart the Url changes to:
  http://localhost:8080/BookstoreWeb/cart/delete/Duobam
  Payment button: It is located at the bottom center of the page.

- **PAYMENT PAGE**
  Link: http://localhost:8080/BookstoreWeb/checkout
  Login page link: http://localhost:8080/BookstoreWeb/user?otype=login
  Register page link: http://localhost:8080/BookstoreWeb/user?otype=register
  Checkout: http://localhost:8080/BookstoreWeb/checkout
  The above link of checkout contains account information, shipping address, items in cart, total bill amount and a textbox where the client have to enter their credit or debit card to verify their information.

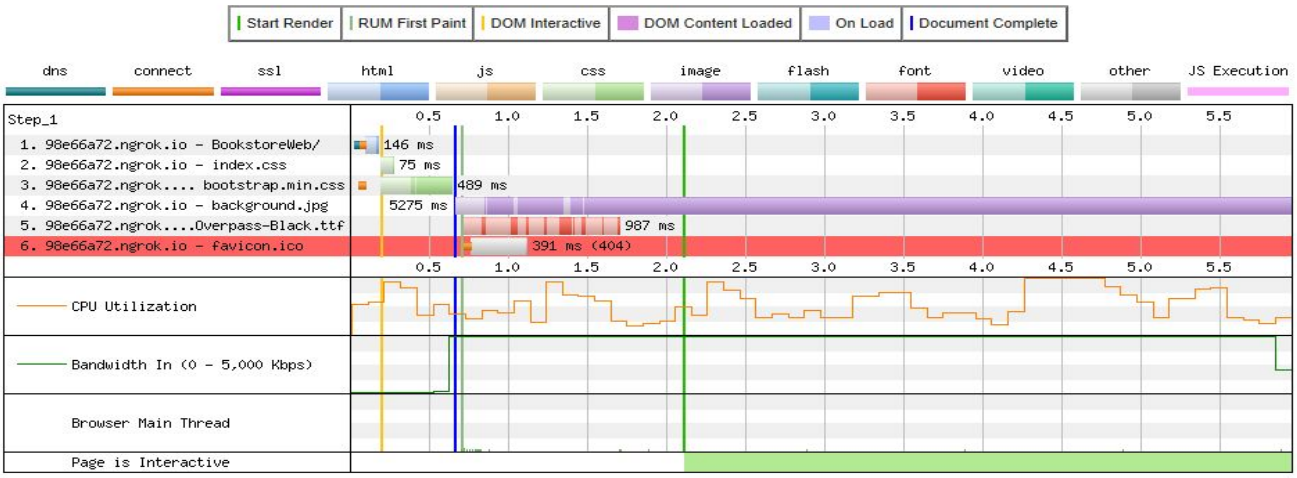
- **ANALYTICS PAGE**
  Report of how many books are sold each month can be found: http://localhost:8080/BookstoreWeb/admin

WEB SERVICES

http://localhost:8080/BookstoreWeb/api/books : GET--get all books in the store
http://localhost:8080/BookstoreWeb/api/books?sort=Science : GET--get book by id
http://localhost:8080/BookstoreWeb/api/books?sort=All : GET--get all books by browsing the category to “all”
http://localhost:8080/BookstoreWeb/api/books?search=%7B%22search%22:%22Duobam%22%7D : GET--search books
http://localhost:8080/BookstoreWeb/cart/proceed : POST--shopping cart
http://localhost:8080/BookstoreWeb/user/login : POST--client login
http://localhost:8080/BookstoreWeb/user/create : POST--register a client
http://localhost:8080/BookstoreWeb/order/create : POST--create order
http://localhost:8080/BookstoreWeb/api/books/reviews : GET--book review
http://localhost:8080/BookstoreWeb/api/books/reviews/add : POST-- post review

PERFORMANCE TESTING REPORT

Waterfall View



Connection View

Request Details

Before Start Render | Before On Load | After On Load

Request Details

#	Resource	Content Type	Request Start	DNS Lookup	Initial Connection	Time to First Byte	Content Download	Bytes Downloaded	Error/Status Code	IP
1	http://98e66a72.ngrok.io/BookstoreWeb/	text/html	0.088 s	26 ms	40 ms	77 ms	3 ms	0.6 KB	200	52.15.72.79
2	http://98e66a72.ngro...oreWeb/css/index.css	text/css	0.193 s	-	-	74 ms	1 ms	1.5 KB	200	52.15.72.79
3	http://98e66a72.ngro...ss/bootstrap.min.css	text/css	0.193 s	-	42 ms	186 ms	261 ms	137.6 KB	200	52.15.72.79
4	http://98e66a72.ngro...ssets/background.jpg	image/jpeg	0.665 s	-	-	185 ms	5090 ms	2,859.0 KB	200	52.15.72.79
5	http://98e66a72.ngro...s/Overpass-Black.ttf	font/ttf	0.711 s	-	-	115 ms	872 ms	105.7 KB	200	52.15.72.79
6	http://98e66a72.ngrok.io/favicon.ico	-	0.758 s	-	41 ms	350 ms	-	-	404	52.15.72.79

TEAM MEMBER CONTRIBUTION

For this project our team worked together on every component of the project. We have use bitbucket to share our updated work and to solve any problem or to make decision on further process we have met and discussed during every wednesday lab hours. We all are grateful for working together on this project; that we had learnt a lot from each other during entire term.

Maitry Patel:

As our team worked together for this project, most of the implementation part was done together as well. Particularly, I have worked on validation for front end as well as server side. For front end validation, I have handled possible errors by using JavaScript. As for concern with server side, validation generated by me are in DAO classes. Throughout the project, modification to JavaScript was constantly required. I also took in consideration security and encryption of client's information such as password. To protect the password, we decided to implement salting algorithm. I have learned through this project is how important error handling and security is important while doing web development.

Komal Patel:

It was great working with my team as everyone was eager to contribute. My main focus for the project was data layer. We decide to use our own schema instead of the schema that was provided to us such that it fitted our design really well.My job was to provide required entities, finding appropriate relationship between them and an interface, bean and DAO classes, and performing updates accordingly. I was able to gain knowledge how data layer works and its contribution to other layers.

Karan Pavra:

We all contribute for this project. But for a project, my major contribution regarding front end development. My work included designing webpage layout and link those to the project's other components using REST services. I focused on technical aspects of the website along with the visual aspect of the website. As the project was progressing, constant updates and additions were required that I have managed to handle. This project has helped me gain understanding of how a website works and designing a website enhances the overall look of the website.

SIGNATURES:

Maitry Patel [215296510], Komal Patel [215306129], Karan Pavra [214025704].

Delivered on:
15th April, 2019