CUSTOM GUITAR BUILDER

Made by:

Palak Sharma (RA1911033010112)

and

Shivang Bhatnagar (RA19119033010119)

INDEX

List of Figures

1. Problem Description

The music industry has a huge influence on people of all ages in these times. With so many talented musicians with their own signature instruments with their own specific quirks and features, people who know how to play different instruments, a custom guitar of your own specifications and design, completely designed by you and “Made in India” is surely going to leave a positive mark for our future generations. Our “Custom Guitar Builder” allows you to design your dream guitar or bass with our online configurator. Select a preset to get started and then choose from a wide array of specs, like body/headstock shape, type of woods, finish colors, binding, and much more.

It’s going to be as simple as choosing what you want on your guitar, from body shape, wood used, to custom paintjob and even a custom headstock of your choice.

2 Software Requirements Specification

2.1. Introduction

2.1.1. Purpose

The purpose of this application is to give customers personalized experience with wide variety of customization options. It is designed to save time of a customer and it is cost effective.

2.1.2. Document Conventions

N/A

2.1.3. Indented Audience and Reading Suggestion

This document should be read by developers, users, project managers and testers. The developers should read every section to ensure that there is an understanding of the project. The main sections for the customers to review are section 2.1.4 Project Scope, 2.2.7 Assumptions and section 2.4 Features.

2.1.4. Project Scope

The main aim of this project is to give customer personal experience and letting customers decide the colour, wood, material, shape etc. will helps them in selecting all those specific things which they wanted in one their guitar to have.

2.1.5. References

None

2.2. Overall Description

2.2.1. Product Perspective

The custom guitar builder provides a simple mechanism for the user to customize and create guitars and basses. The application does not require the user to sign in to their account every time. It allows the user to customize and create a new guitar and place an order.

2.2.2. Product Functions

The Application includes a range of functions that enables the user customize and create guitars and basses.

i) General options: dexterity, construction method, no. of strings and scale length.

ii) Body: shape, contour, wood, colour, custom graphics and finish.

iii) Neck: neck wood, fretboard wood, fretboard radius, frets, inlays and bindings.

iv) Headstock: shape, angle and truss cover.

v) Components: hardware colour, bridge, tuners, pick guard, pickups, control pattern, knobs, cavity covers and switch caps.

vi) Special options: chambered body, strap locks, kill switch, coil split, locking output jack, your signature on headstock and stainless-steel frets.

2.2.3. User Classes and Characteristics

Any adult who possess a compatible smart phone will be comprise the user audience. Musician who are looking for an Indian made custom guitar want will be the target demographic.

2.2.4. Operating Environment

The Application will operate in the following operating environment:
Android OS iOS.

2.2.5. Design and Implementation Constrains

The Application's Android variant is created using Java programming language and the Android API. So, the Android variant is compatible with android devices running Android 6 or above with a minimum RAM of 2 GB. The Application's iOS variant is created using Objective C++ programming language and the iOS API. So, the iOS variant is compatible with iOS devices running iOS 8 or above. For Language support expect from the basic English language pack the user can download and enable the language pack of their choice from the list of available languages within the application. For connection stream TCP/IP is used as it is the common gateway for internet applications.

2.2.6. User Documentation

There will be a basic tutorial document along with an in-app tutorial to aid users.

2.2.7. Assumptions and Dependencies

i) The app remains stable and compatible with Android 6.0 and greater.

ii) The app will be completely functional.

2.3. External Interface Requirements

2.3.1. User Interface

The look and feel must be simple and elegant for users to like it. The app will follow the colour code of the desktop website of the application. The font size is appropriate and the currency symbols are synchronized.

2.3.2. Hardware Interface

The app primarily runs on smart phones. So, the interface through which the user interacts should be touch enabled. For the communication purpose, the program needs these protocols to be installed:

TCP for the client to connect to the server in online mode.

2.3.3. Software Interface

The software interface will be Android and iOS.

2.3.4. Communication Interface

Setting up the server into server mode requires that there will be open ports for accepting connections from the clients. The connection between the client and the server uses connection-oriented communication, via TCP/IP - Transfer Control Protocol / Internet Protocol, implements reliable delivery of messages. Connection oriented communication makes programming easier because the protocol includes mechanisms for detecting and handling error and an acknowledgement mechanism between client and server.

2.4 System Features

2.4.1 General Options

i) Dexterity (left-handed or right-handed)

ii) Construction methods (bolt on neck, set neck, neck through)

iii) Number of strings (6,7,8)

iv) Scale length (multi scale, common scale)

2.4.2 Body

i) Shape (shape of your choice)

ii) Contour (flat, tapered edges, curved carved)

iii) Wood (top wood, core wood, back wood)

iv) Colour (colour of your choice)

v) Custom graphics and finish (of your choice)

2.4.3 Neck

i) Wood (wood of your choice)

ii) Fretboard wood (wood of your choice)

iii) Fretboard radius (cheap or flat)

iv) Frets (no. of frets)

v) Inlays (centre dots, offset dots, blocks, shark fins, of your choice)

vi) Binding (white, green, black, of your choice)

2.4.4 Headstock

i) Shape (shape of your choice)

ii) Angle (straight or angled)

iii) Truss cover (black, white, cream, of your choice)

2.4.5 Components

i) Hardware colour (chrome, black, gold)

ii) Bridge (of your choice)

iii) Tuners (locking, non-locking)

iv) Pick guard (of your choice)

v) Pickups (active, passive, configuration)

vi) Control pattern

vii) Knobs (volume and tone knobs of your choice)

vii) Cavity covers (black, white, cream)

viii) Switch caps (black, white, cream)

2.4.6 Special options

i) Chambered body

ii) Strap locks

iii) Kill switch

iv) Coil split

v) Locking output jack

vi) Your signature on headstock

vii) Stainless steel frets

viii) Special instruction (any special instruction)

USE CASE DIAGRAM

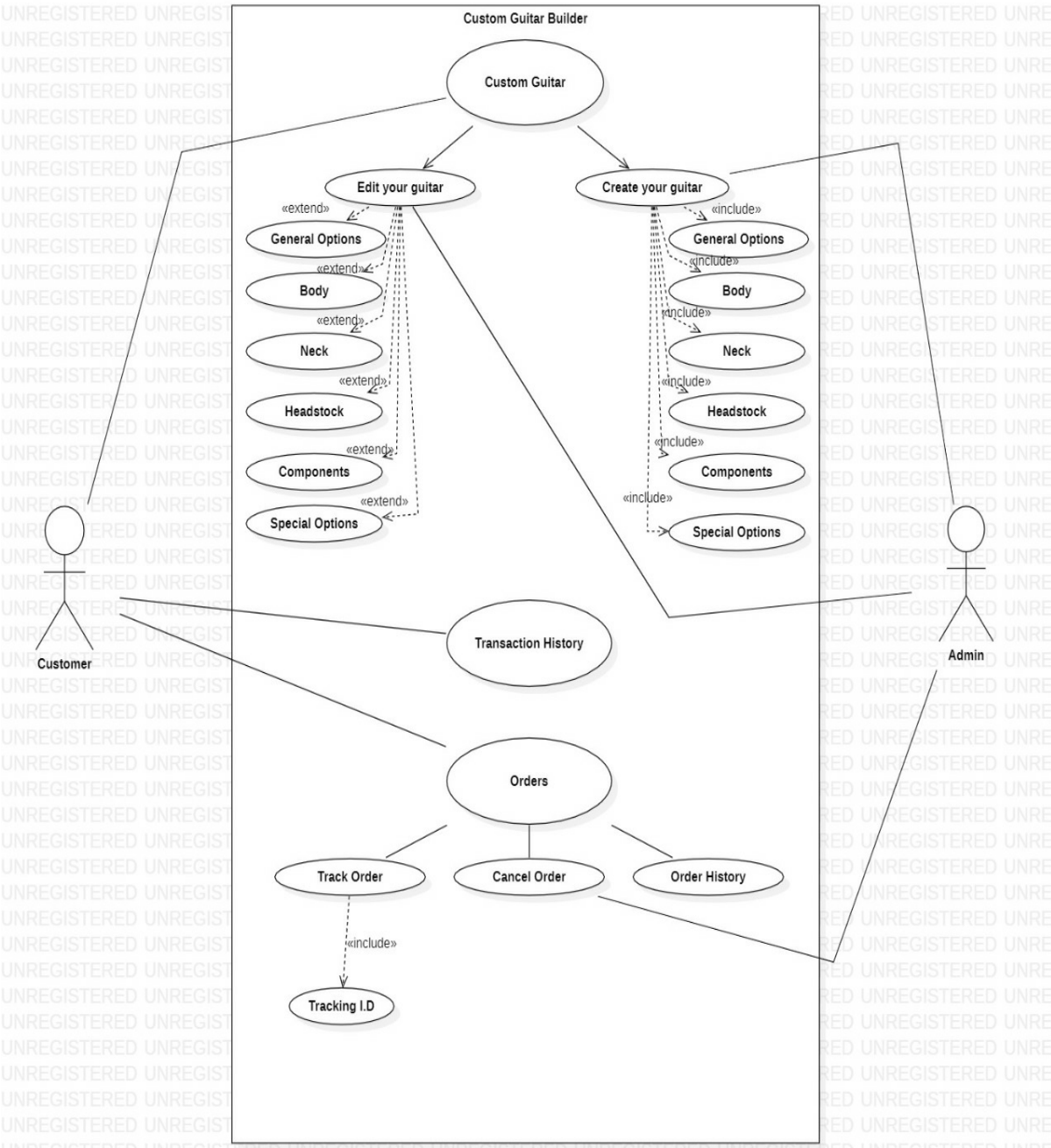

Fig 3.1 Use Case Diagram

3.1 Use Case Documentation

Custom guitar Use Case Diagram

Use case name	Custom guitar
Actor	Customer
Pre-condition	Must be logged in
Post-condition	*nil*
Include	*nil*
Extent	*nil*
Frequency of use	Frequently
Normal course of events	User chooses to either edit his or her guitar(s) or create a new one.

Create your guitar Use Case Diagram

Use case name	Create your guitar
Actor	Customer, Admin
Pre-condition	Must be logged in
Post-condition	*nil*
Include	*Six*
Extent	*nil*
Frequency of use	Frequently
Normal course of events	User chooses to create a new guitar.

Edit your guitar Use Case Diagram

Use case name	Edit your guitar
Actor	Customer, Admin
Pre-condition	Must be logged in
Post-condition	*nil*
Include	*nil*
Extent	Six
Frequency of use	Less frequently
Normal course of events	User chooses to edit his or her guitar(s).

Transaction History Use Case Diagram

Use case name	Transaction History
Actor	Customer
Pre-condition	Must be logged in
Post-condition	*nil*
Include	*nil*
Extent	*nil*
Frequency of use	Less frequently
Normal course of events	Users can see their transaction history

Orders Use Case Diagram

Use case name	Orders
Actor	Customer
Pre-condition	Must be logged in
Post-condition	*nil*
Include	*nil*
Extent	*nil*
Frequency of use	Frequently
Normal course of events	Users can see their order history, cancel an order and track an order

Track Order Use Case Diagram

Use case name	Track Order
Actor	Customer
Pre-condition	Must be logged in
Post-condition	*nil*
Include	one
Extent	*nil*
Frequency of use	Frequently
Normal course of events	Users can track their order

Cancel Order Use Case Diagram

Use case name	Cancel Order
Actor	Customer, Admin
Pre-condition	Must be logged in
Post-condition	*nil*
Include	*nil*
Extent	*nil*
Frequency of use	Less Frequently
Normal course of events	Users can cancel their order.

Use Case Diagram

Use case name	Order History
Actor	Customer
Pre-condition	Must be logged in
Post-condition	*nil*
Include	*nil*
Extent	*nil*
Frequency of use	Less Frequently
Normal course of events	Users can see their order history.

3.2 Use Case Checklist

Sl. No.	Conventions to be Checked	Use Case Checklist							
		Custom Guitar	Edit your Guitar	Create your guitar	Transaction History	Orders	Track Order	Cancel Order	Order History
1	Use Case Names Begin With a Strong Verb	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	Name Use Cases Using Domain Terminology	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3	Place Your Primary Use Cases In The Top Left Corner Of The Diagram	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	Imply Timing Consideration By Stacking Use Cases	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	Use Case Names Use An Active Voice	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6	Use Case Names Use The Present Tense	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Sl. No.	Conventions to be Checked	Use Case Checklist							
		Custom Guitar	Edit your guitar	Create your guitar	Transaction History	Orders	Track Order	Cancel Order	Order History
1	Place Your Primary Actor(S) In The Top-Left Corner Of The Diagram.	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	Name Actors With Singular, Business-Relevant Nouns	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3	Associate Each Actor With One Or More Use Cases	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	Actors Model Roles, Not Positions	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	Use <<system>> to Indicate System Actors	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6	Actors Don't Interact With One Another	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Sl. No.	Conventions to be Checked	Use Case Check list							
		Custom Guitar	Edit your Guitar	Create your Guitar	Transaction History	Orders	Track Order	Cancel Order	Order History
1	Indicate An Association Between An Actor And A Use Case If The Actor Appears Within The Use Case Logic	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2	Avoid Arrowheads On Actor-Use Case Relationships	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3	Apply <<include>> When You Know When To Invoke The Use Case	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4	Apply <<extend>> When A Use Case May Be Invoked Across Several Use Case Steps	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5	Introduce <<extend>> Associations sparingly	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6	Generalize Use Cases When A Single Condition Results Significantly New Business Logic	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
7	Avoid More Than Two Levels Of Use Case Associations	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
8	Place An Included Use Case To The Right Of The Invoking Use Case	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
9	Place An Extending Use Case Below The Parent Use Case	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
10	Place an Inheriting Use Case Below The Base Use Case	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

CLASS DIAGRAM

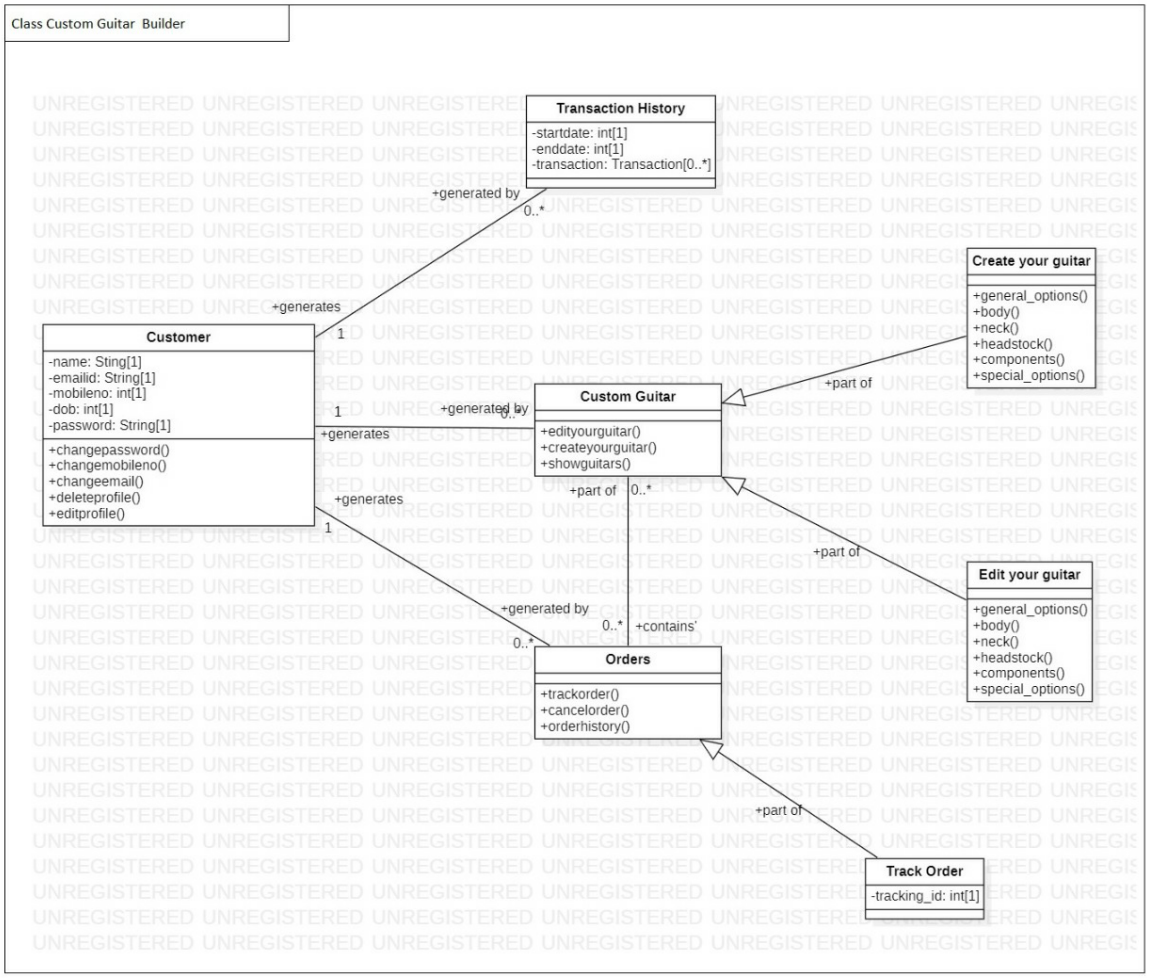

Fig 4.1 Class Diagram

4.2 Class Checklist

Sl. No.	Parameters	Class Checklist(Y/N)						
		Customer	Transaction History	Custom Guitar	Orders	Create your guitars	Edit your guitars	Track Order
1.	Put common terminology for names	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2.	Choose complete singular nouns over class names	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3.	Names operations with a strong verb	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4.	Names attributes with a domain-based noun	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5.	Never show classes with just two compartments	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6.	Label uncommon class compartments	Yes	Yes	Yes	Yes	Yes	Yes	Yes
7.	Include an ellipsis (…) at the end of incomplete lists	Yes	Yes	Yes	Yes	Yes	Yes	Yes
8.	List static operations/attributes before instance operations/attributes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
9.	List operations/attributes in decreasing visibility	Yes	Yes	Yes	Yes	Yes	Yes	Yes
10.	For parameters that are objects only list their type	Yes	Yes	Yes	Yes	Yes	Yes	Yes
11.	Develop consistent method signatures	Yes	Yes	Yes	Yes	Yes	Yes	Yes
12.	Avoid stereotypes implied by language naming conventions	Yes	Yes	Yes	Yes	Yes	Yes	Yes
13.	Indicate exceptions in an operation's property string.	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Sl. No.	Parameters	Class Checklist(Y/N)						
		Customer	Transaction history	Custom Guitar	Orders	Create your Guitar	Edit your Guitar	Track Order
1.	Interface definitions must reflect implementation language constraints	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2.	Name interfaces according to language naming conventions	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3.	Apply "Lollipop" notations to indicate that a class realizes an interface	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4.	Define interfaces separately from your classes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5.	Do not model the operations and attributes of an interface in your classes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6.	Consider an interface to be a contract	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Sl. No.	Parameters	Class Checklist(Y/N)						
		Customer	Transaction History	Custom Guitar	Orders	Create your guitars	Edit your guitar	Track Order
1.	Put in the sentence rule for inheritance	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2.	Put subclasses below super classes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3.	Ensure that you are aware of data-based interface	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4.	A subclass must inherit everything	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Sl. No.	Parameters	Class Checklist(Y/N)						
		Customer	Transaction history	Custom Guitar	Orders	Create your guitar	Edit your guitars	Track Orders
1.	Ensure that you model relationships horizontally	Yes	Yes	Yes	Yes	Yes	Yes	Yes
2.	Collaboration means a need for a relationship	Yes	Yes	Yes	Yes	Yes	Yes	Yes
3.	Model a dependency when a relationship is in transition	Yes	Yes	Yes	Yes	Yes	Yes	Yes
4.	Depict similar relationships involving a common class as a tree	Yes	Yes	Yes	Yes	Yes	Yes	Yes
5.	As a rule it is best to always indicate the multiplicity	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6.	Avoid multiplicity "*" to avoid confusion	Yes	Yes	Yes	Yes	Yes	Yes	Yes
7.	Replace relationships by indicating attribute types.	Yes	Yes	Yes	Yes	Yes	Yes	Yes
8.	Never model implied relationships	Yes	Yes	Yes	Yes	Yes	Yes	Yes
9.	Never model every single dependency	Yes	Yes	Yes	Yes	Yes	Yes	Yes
10.	Center names on associations	Yes	Yes	Yes	Yes	Yes	Yes	Yes
11.	Write concise association names in active voice	Yes	Yes	Yes	Yes	Yes	Yes	Yes
12.	Indicate directionality to clarify an association name	Yes	Yes	Yes	Yes	Yes	Yes	Yes
13.	Name unidirectional associations in the same directions	Yes	Yes	Yes	Yes	Yes	Yes	Yes
14.	Word association names left-to-right	Yes	Yes	Yes	Yes	Yes	Yes	Yes
15.	Indicate role names when multiple associations between two classes exist	Yes	Yes	Yes	Yes	Yes	Yes	Yes
16.	Indicate role names on recursive associations	Yes	Yes	Yes	Yes	Yes	Yes	Yes
17.	Make associations bi-directional only when collaboration occurs in both directions	Yes	Yes	Yes	Yes	Yes	Yes	Yes
18.	Redraw inherited associations only when something changes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
19.	Questions multiplicities involving minimums and maximums	Yes	Yes	Yes	Yes	Yes	Yes	Yes

SEQUENCE DIAGRAM

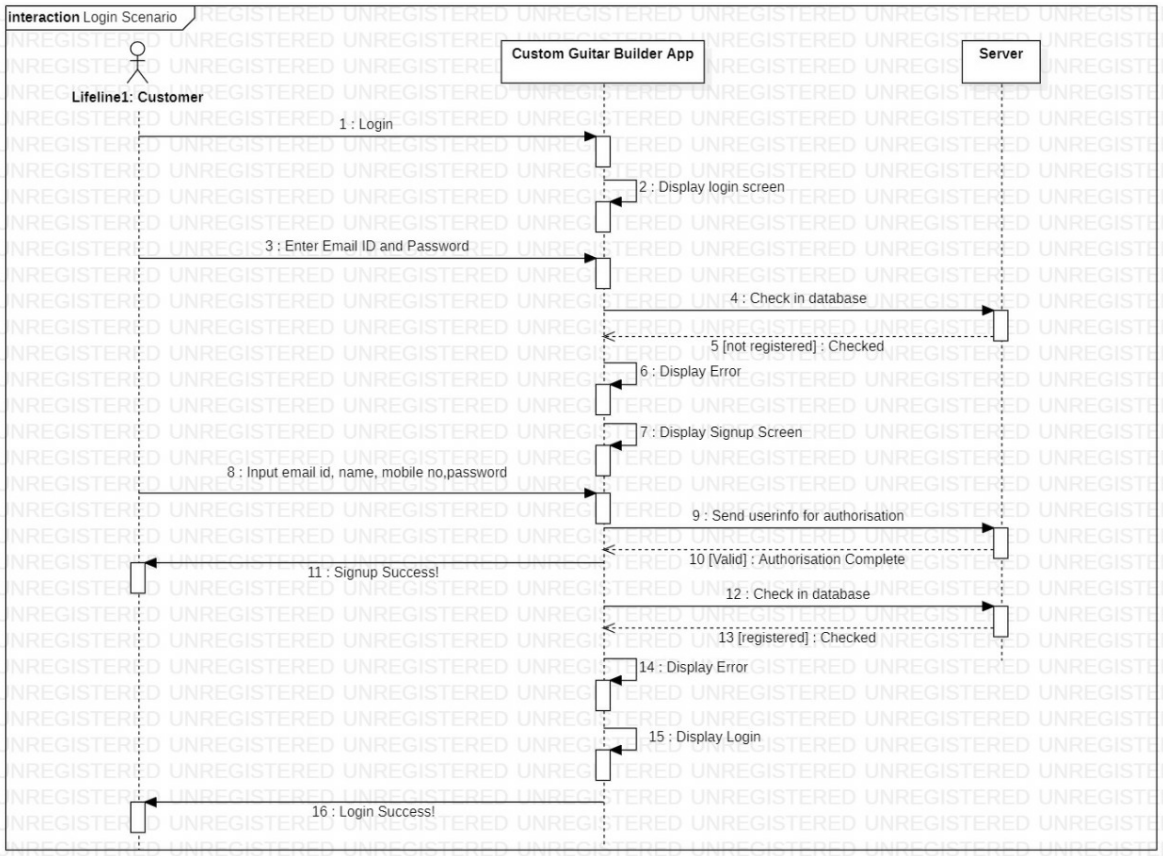
interaction Login Scenario
Custom Guitar Builder App
Server
Lifeline1: Customer
1 : Login
2 : Display login screen
3 : Enter Email ID and Password
4 : Check in database
5 [not registered] : Checked
6 : Display Error
7 : Display Signup Screen
8 : Input email id, name, mobile no,password
9 : Send userinfo for authorisation
10 [Valid] : Authorisation Complete
11 : Signup Success!
12 : Check in database
13 [registered] : Checked
14 : Display Error
15 : Display Login
16 : Login Success!

Fig 5.1

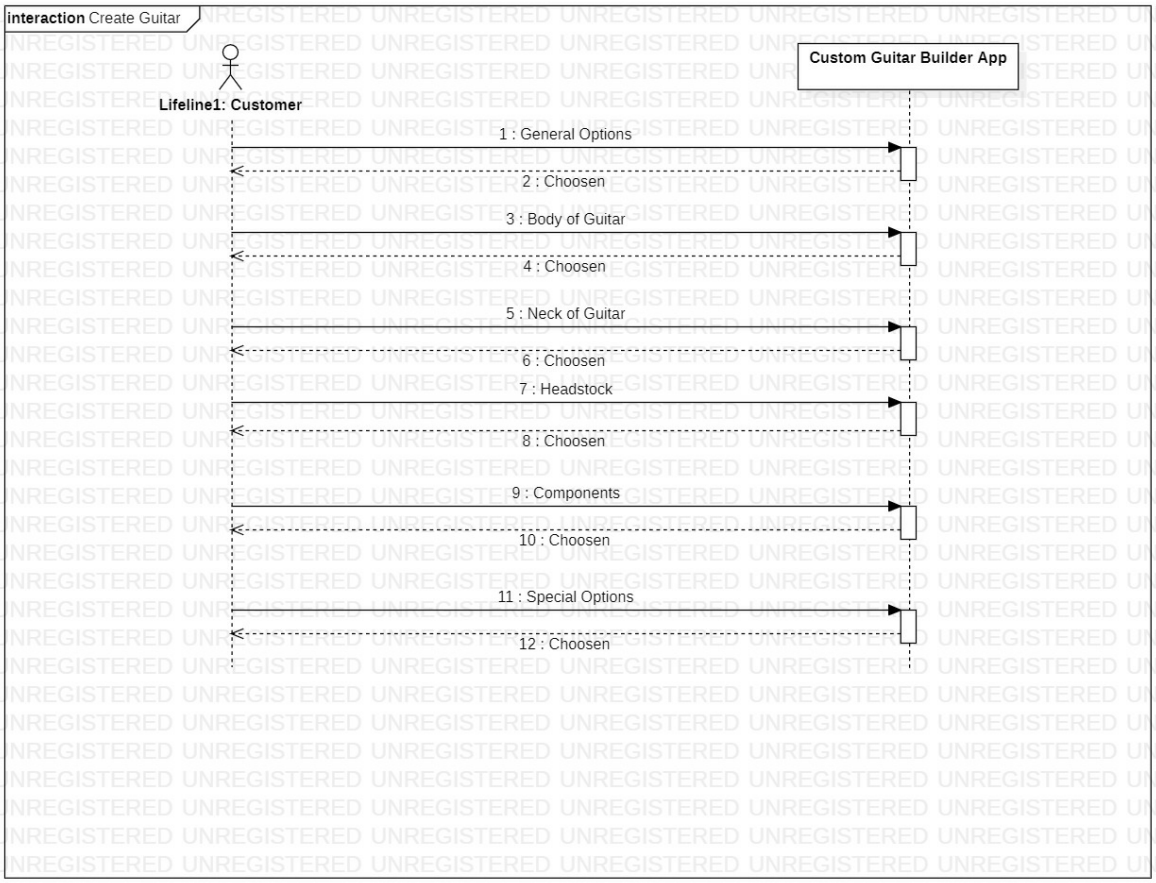
interaction Create Guitar
Custom Guitar Builder App
Lifeline1: Customer
1 : General Options
2 : Choosen
3 : Body of Guitar
4 : Choosen
5 : Neck of Guitar
6 : Choosen
7 : Headstock
8 : Choosen
9 : Components
10 : Choosen
11 : Special Options
12 : Choosen

Fig 5.2

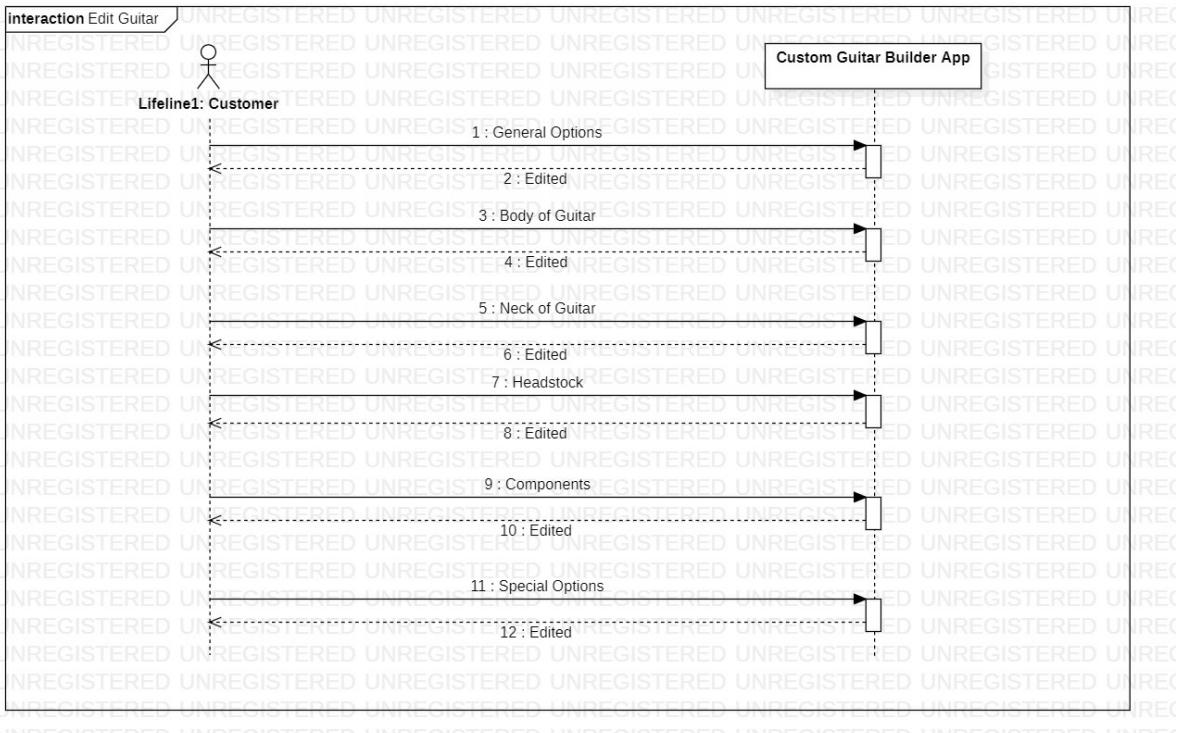
interaction Edit Guitar
Custom Guitar Builder App
Lifeline1: Customer
1 : General Options
2 : Edited
3 : Body of Guitar
4 : Edited
5 : Neck of Guitar
6 : Edited
7 : Headstock
8 : Edited
9 : Components
10 : Edited
11 : Special Options
12 : Edited

Fig 5.3

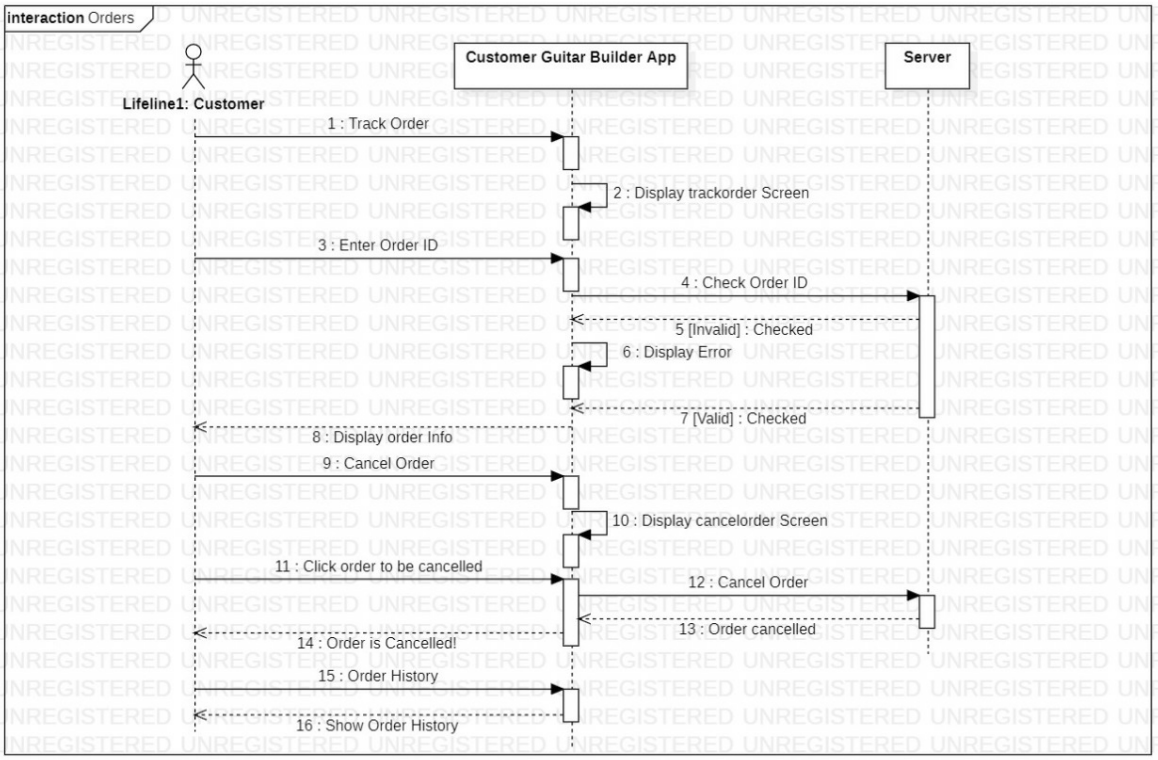

Fig 5.4

5.5. Sequence Checklist

S. No.	Sequence Checklist	Check(Y/N)
1	Messages are from Left-To-Right	Yes
2	Actors named consistently with Use Case Diagram	Yes
3	Classes named consistently with class Diagram	Yes
4	Human and Organisation actors on left most side	Yes
5	Reactive system actors on right most side	Yes
6	Proactive system actors on left most side	Yes
7	Message names beside arrowhead justified	Yes
8	Do not return value when it is obvious	Yes
9	Use return value only when you need to refer it elsewhere	Yes

COMMUNICATION DIAGRAM

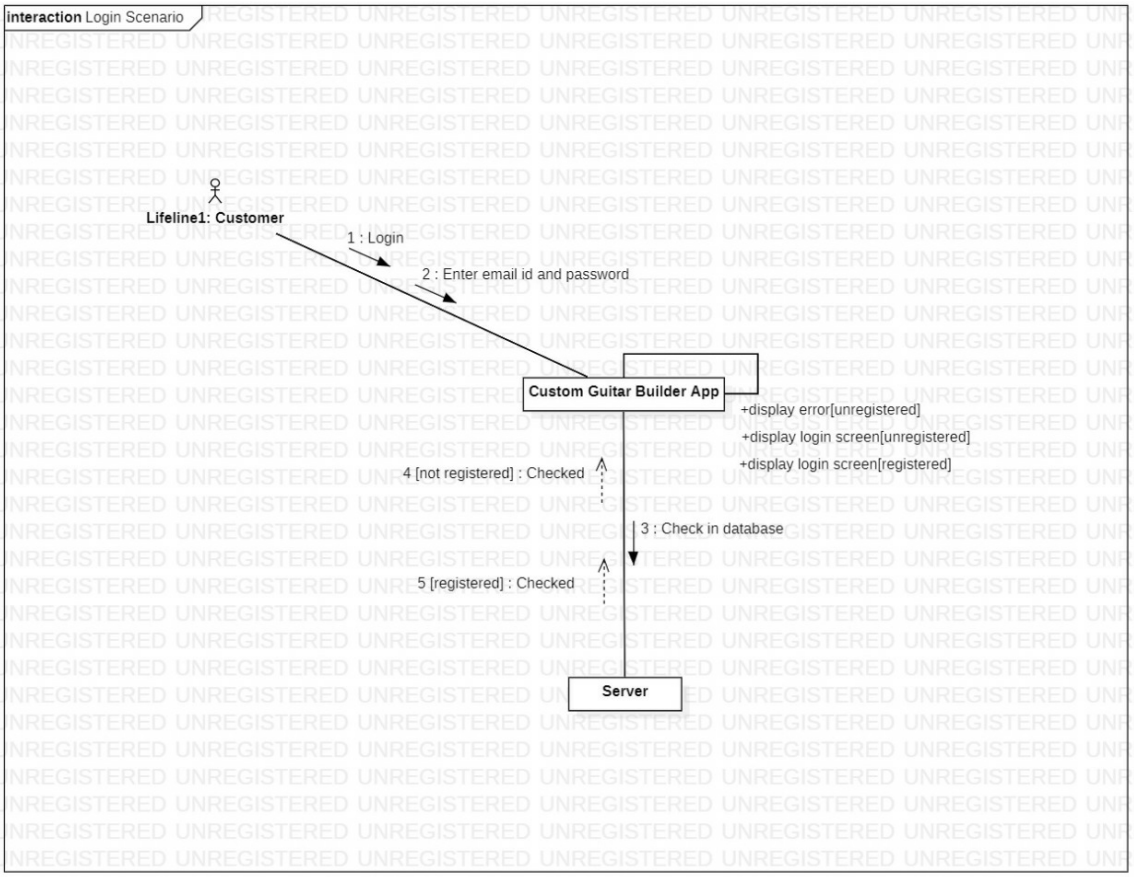
interaction Login Scenario
Lifeline1: Customer
1 : Login
2 : Enter email id and password
Custom Guitar Builder App
+display error[unregistered]
+display login screen[unregistered]
+display login screen[registered]
4 [not registered] : Checked
3 : Check in database
5 [registered] : Checked
Server

Fig 6.1

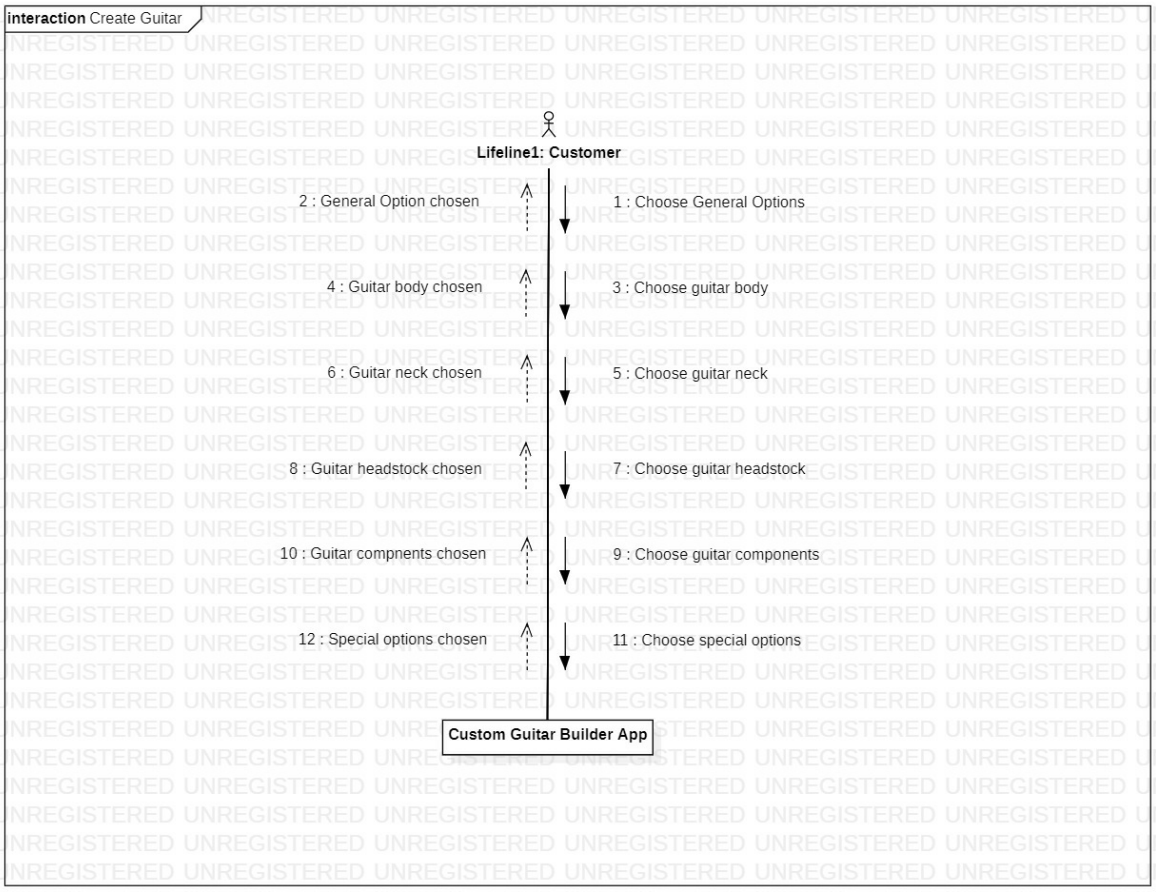
interaction Create Guitar
Lifeline1: Customer
2 : General Option chosen
1 : Choose General Options
4 : Guitar body chosen
3 : Choose guitar body
6 : Guitar neck chosen
5 : Choose guitar neck
8 : Guitar headstock chosen
7 : Choose guitar headstock
10 : Guitar compnents chosen
9 : Choose guitar components
12 : Special options chosen
11 : Choose special options
Custom Guitar Builder App

Fig 6.2

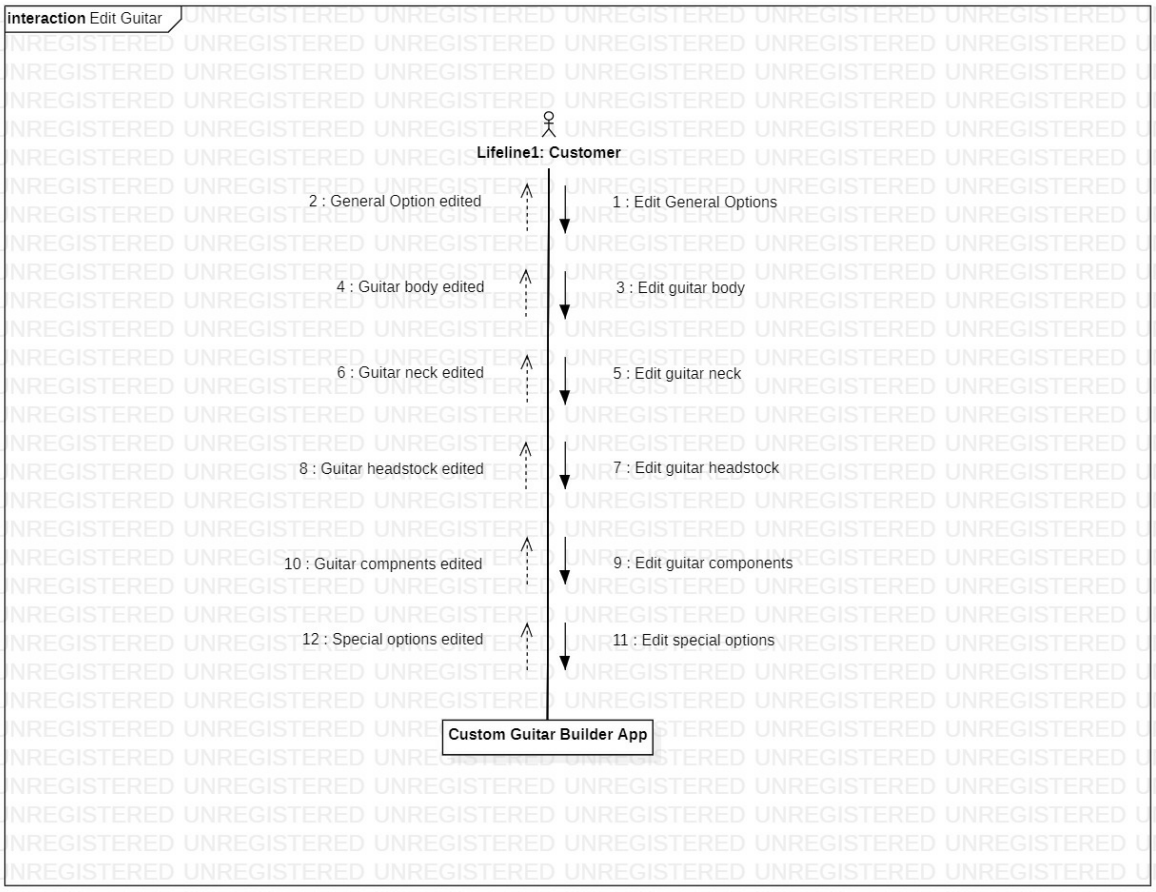
interaction Edit Guitar
Lifeline1: Customer
1 : Edit General Options
2 : General Option edited
3 : Edit guitar body
4 : Guitar body edited
5 : Edit guitar neck
6 : Guitar neck edited
7 : Edit guitar headstock
8 : Guitar headstock edited
9 : Edit guitar components
10 : Guitar compnents edited
11 : Edit special options
12 : Special options edited
Custom Guitar Builder App

Fig 6.3

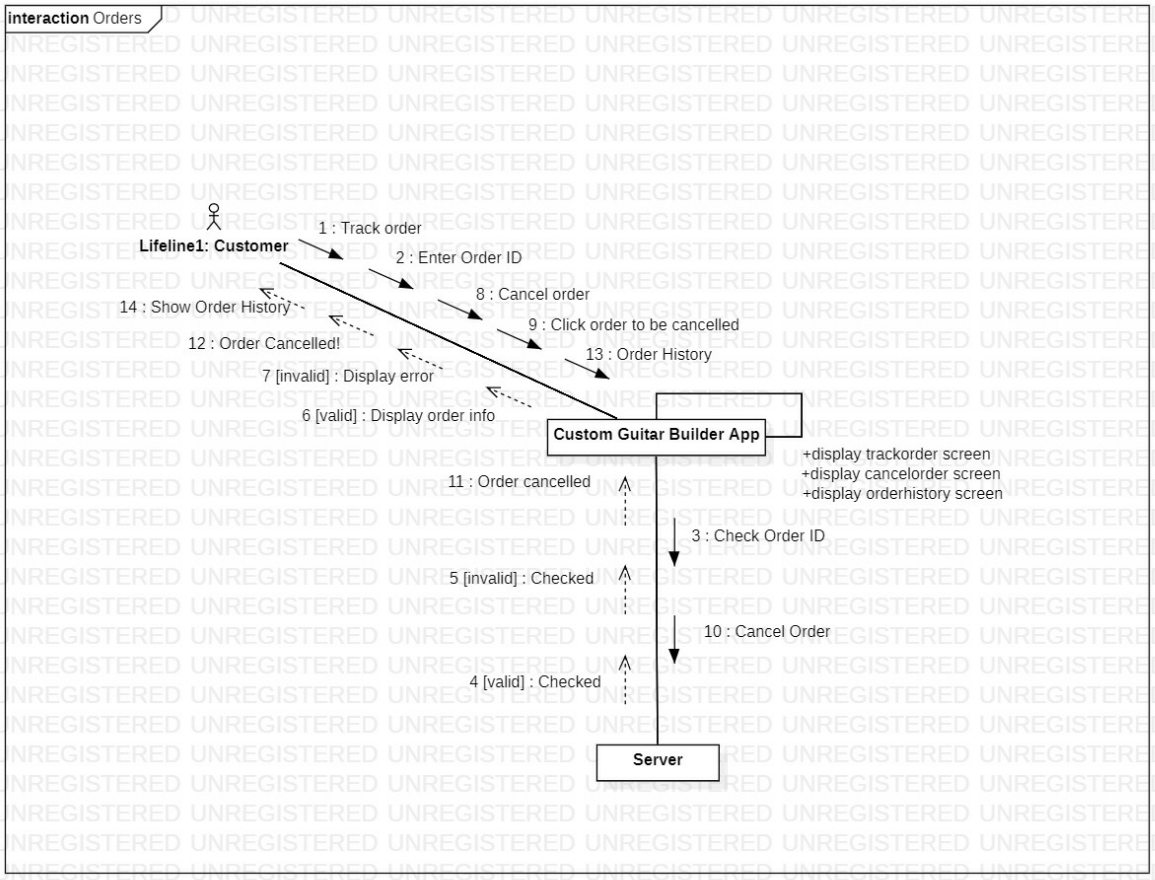

Fig 6.4

6.5. Communication Checklist

S. No.	Communication Checklist	Check(Y/N)
1	Name objects when referred to in messages	Yes
2	Name objects when several of same type exist	Yes
3	Do not model return value when it is obvious	Yes
4	Model return value when you need to refer to it elsewhere	Yes
5	Indicate return value when it is not clear	Yes
6	Indicate parameters when they are not clear	Yes
7	Depict arrow for each message	Yes
8	Indicate navigability sparingly	Yes
9	Prefer roles on links instead of within classes	Yes

BEHAVIOURAL DIAGRAM

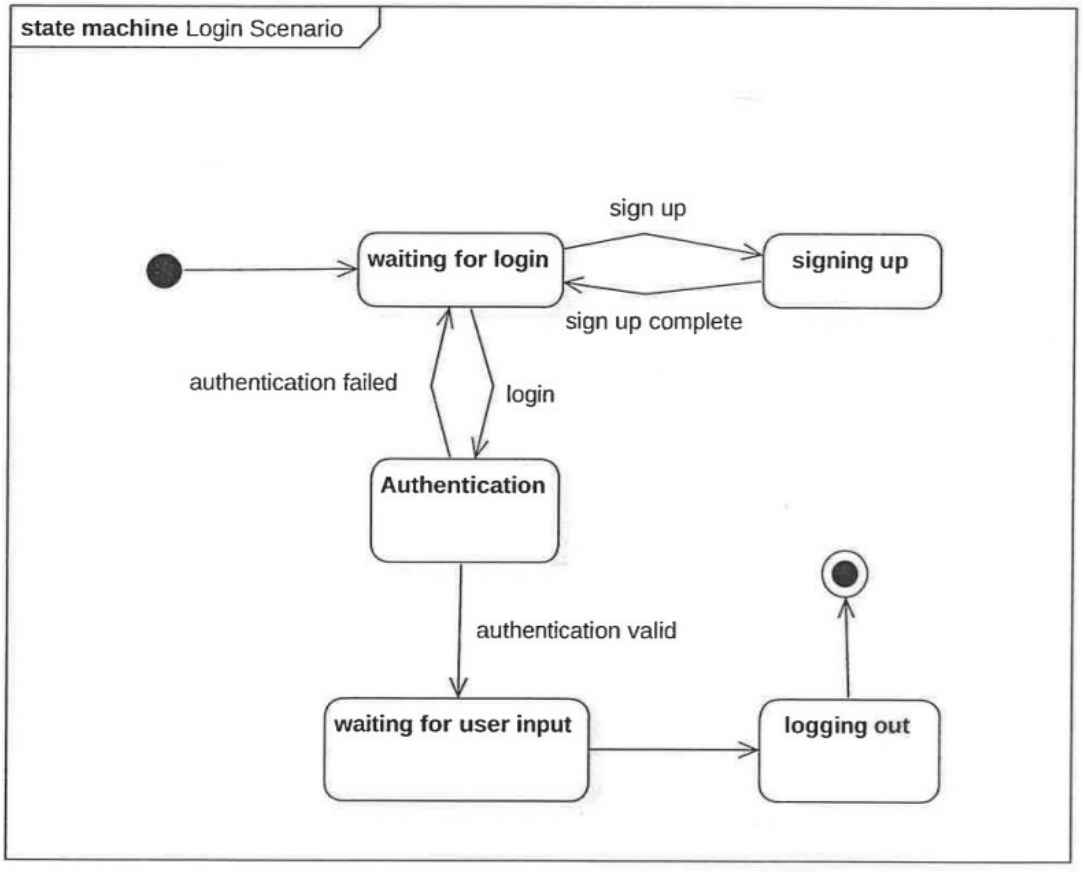
state machine Login Scenario
sign up
waiting for login
signing up
sign up complete
authentication failed
login
Authentication
authentication valid
waiting for user input
logging out

Fig 7.1

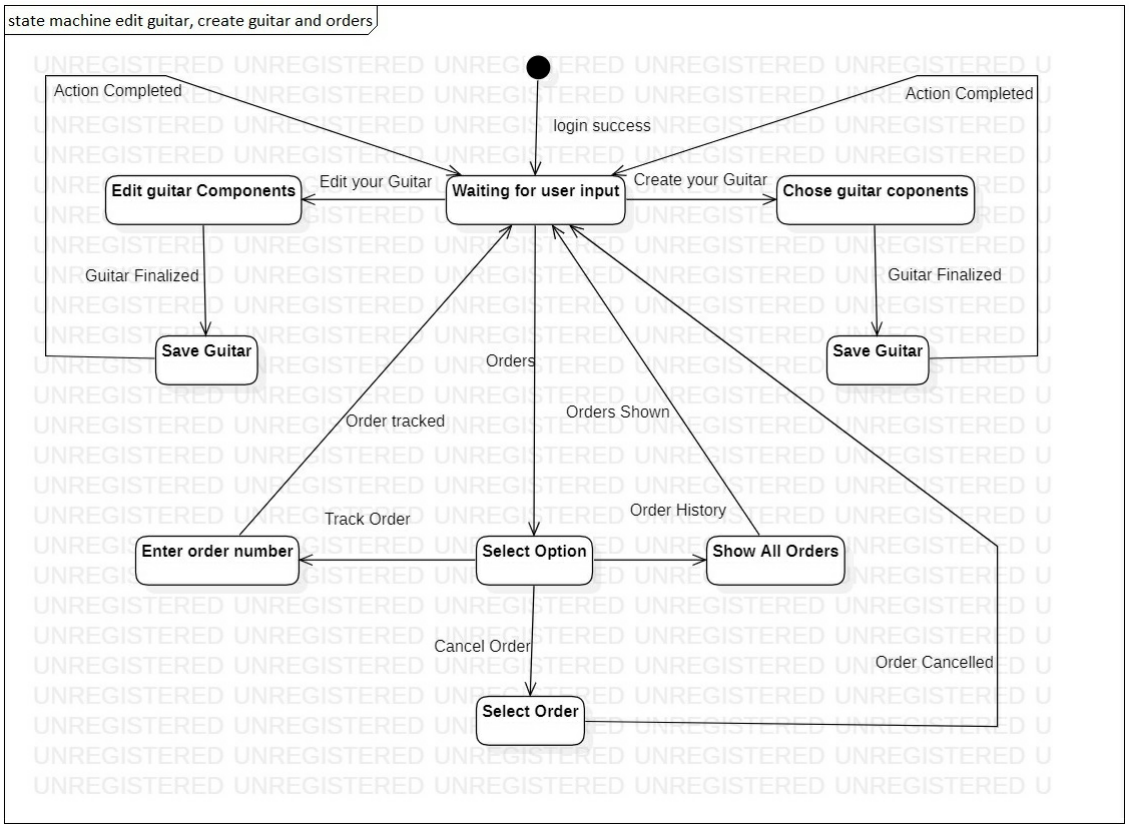

Fig 7.2

7.3 State Chart Checklist

S.NO	State Chart Checklist	Check(Y/N)
1	Initial state on top left corner	Yes
2	Final state on bottom right corner	Yes
3	State names should be simple but descriptive	Yes
4	Model Sub states for targeted complexity	Yes
5	Aggregate common Sub state transitions	Yes
6	Top-level state machines always have initial and final states	Yes
7	Name software actions using implementation language naming conventions	Yes
8	Name actors using prose	Yes
9	Indicate entry actions wherever applicable	Yes
10	Indicate exit actions wherever applicable	Yes
11	Model recursive transitions only when you want to exit and re-enter the state	Yes
12	Name transition event in past tense	Yes
13	Place transition label near source state	Yes
14	Place transition label based on source direction	Yes

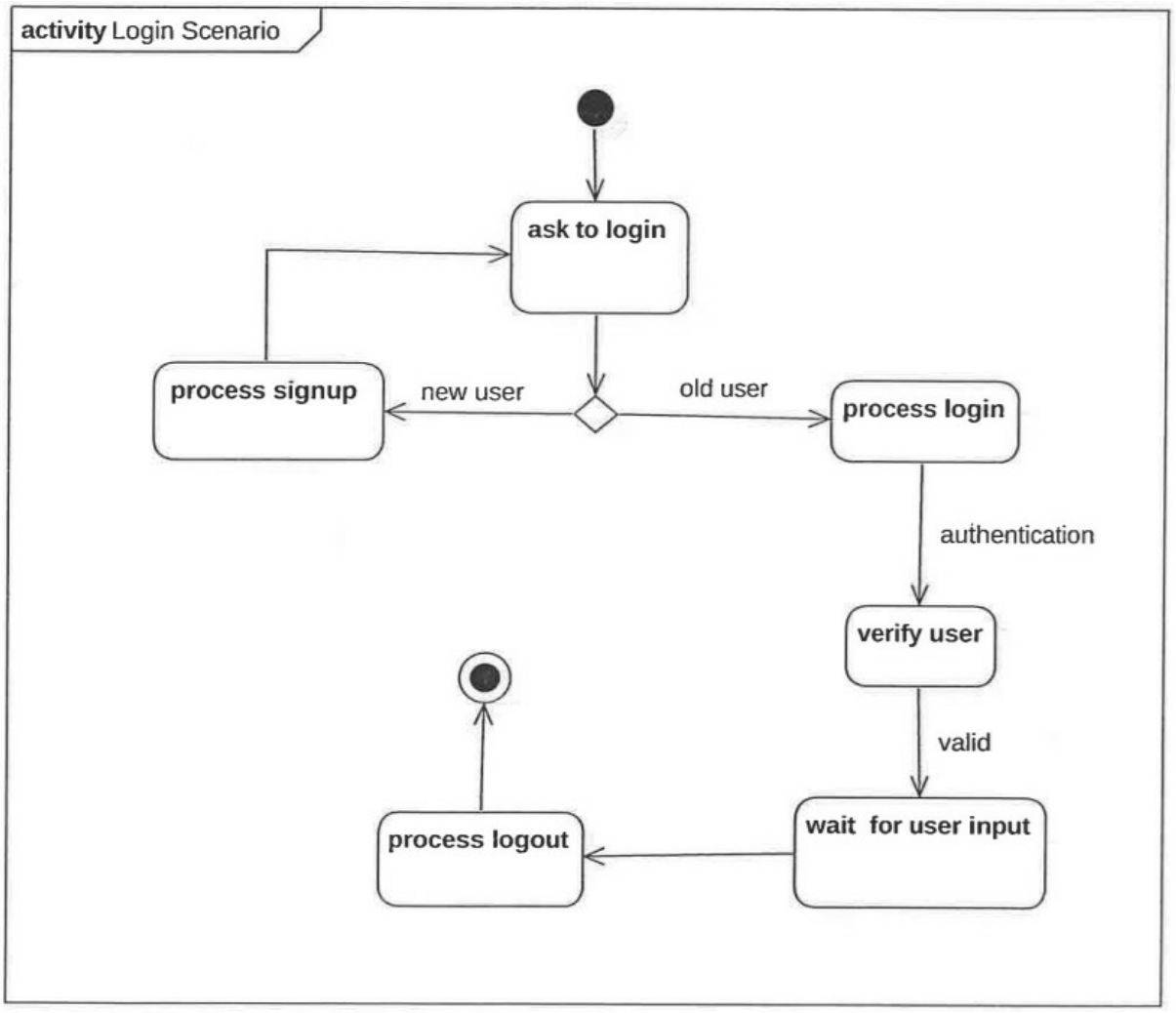
activity Login Scenario
ask to login
process signup
new user
old user
process login
authentication
verify user
valid
wait for user input
process logout

Fig 8.1

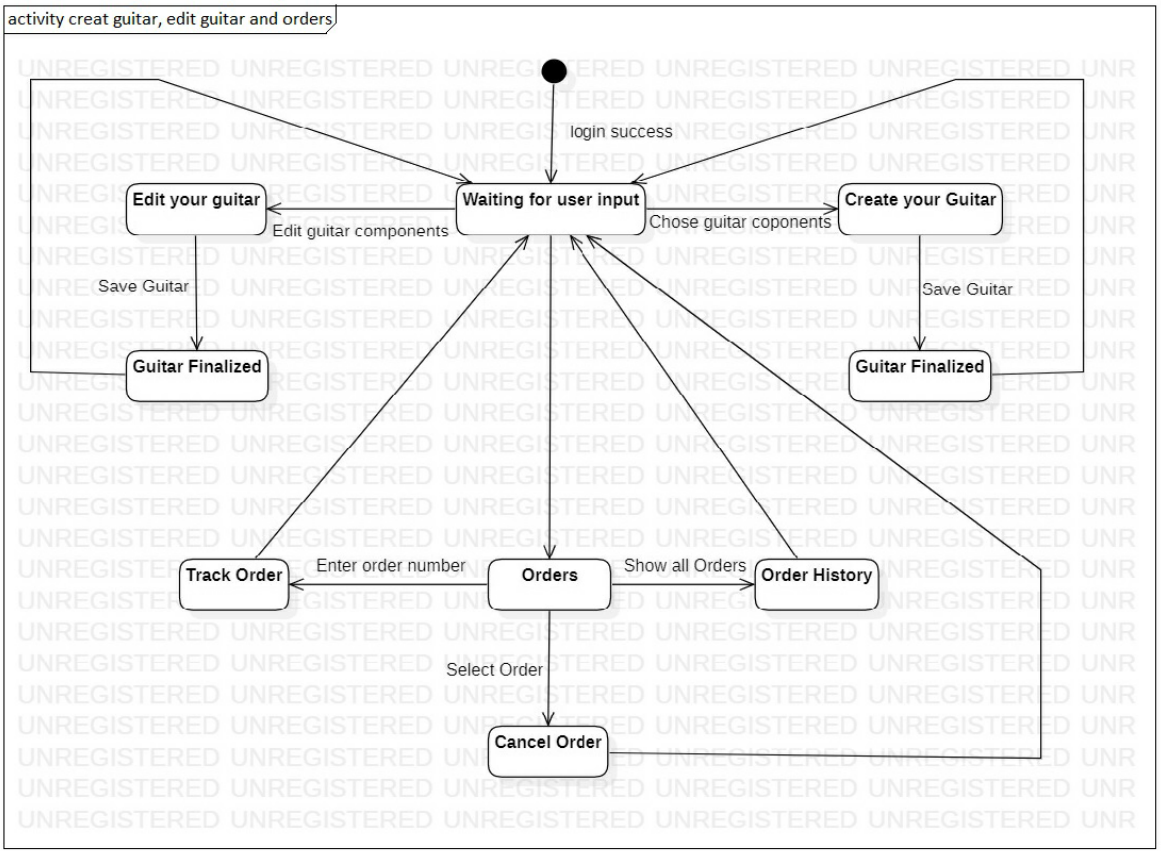

Fig 8.2

8.3 Activity Checklist

S.NO	Activity Checklist	Check(Y/N)
1	Place start point on top left corner	Yes
2	Always include end point	Yes
3	Simplify Flow charting operations	Yes
4	Decision points should reflect previous activity	Yes
5	Avoid Superfluous decision points	Yes
6	Each transition leaving point must have a guard	NIL
7	Guards should not overlap	NIL
8	Guards on decision points must forma complete set	NIL
9	Exit transition guards and Activity Invariants must forma complete set	NIL
10	A fork should have a corresponding join	NIL
11	Forks only have one entry transition	NIL
12	Joins only have one exit transition	NIL
13	Avoid superfluous forks	NIL
14	Order Swim lanes in logical manner	NIL
15	Apply swim lanes to linear processes	NIL
16	Have less than 5 swim lanes	NIL
17	Consider horizontal swim lanes for business processes	

COMPONENT DIAGRAM

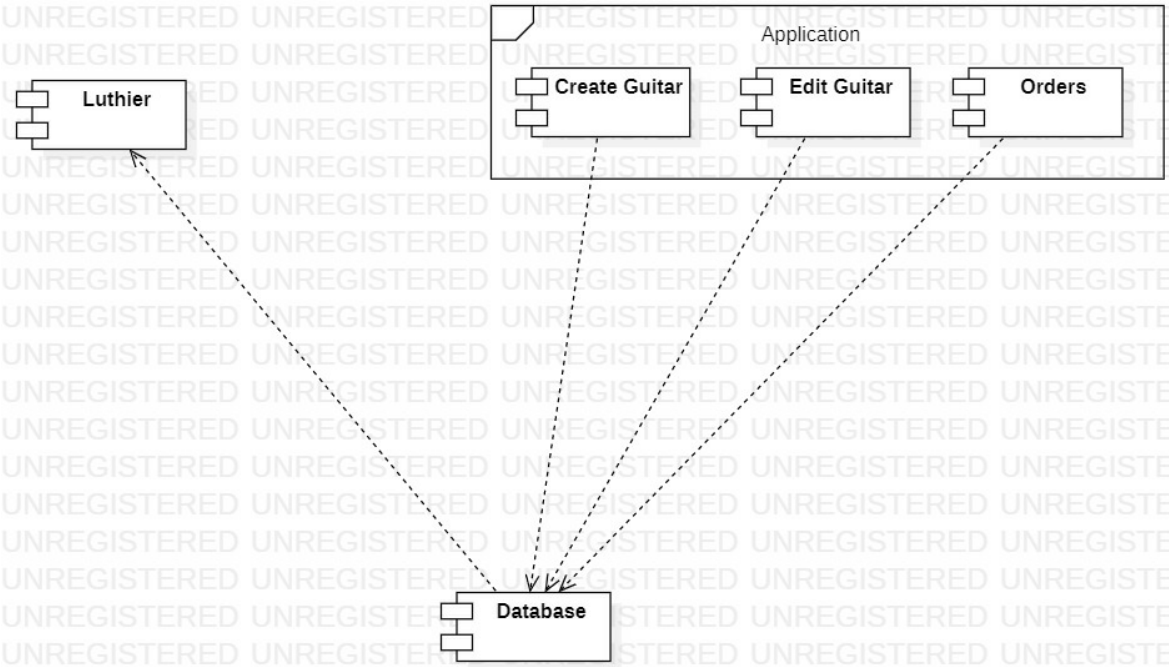
Application
Luthier
Create Guitar
Edit Guitar
Orders
Database

Fig 9.1

Deployment Diagram

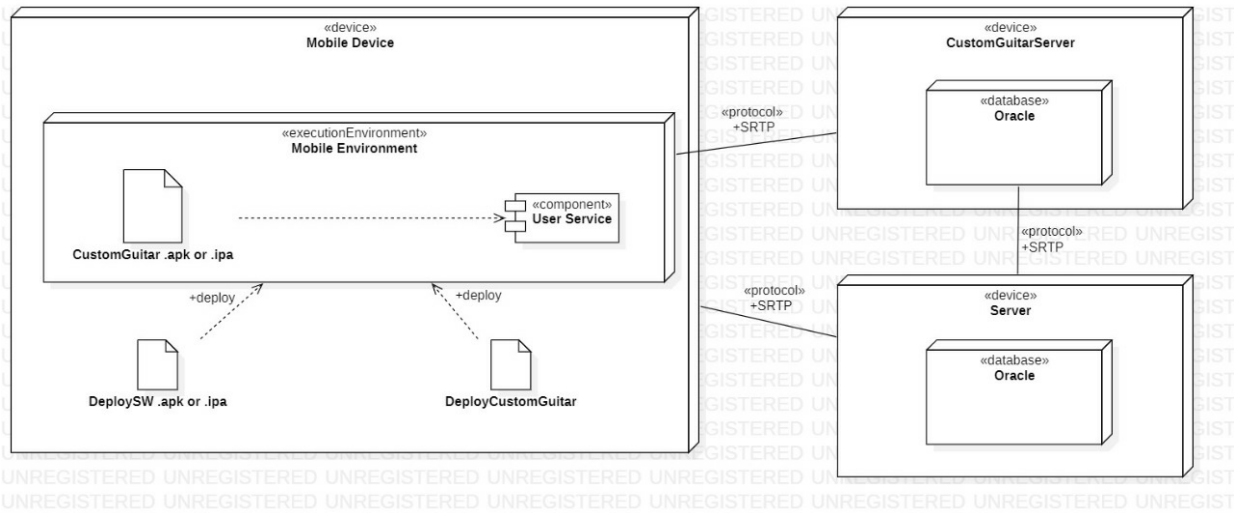
«device»
Mobile Device
«executionEnvironment»
Mobile Environment
CustomGuitar .apk or .ipa
«component»
User Service
+deploy
+deploy
DeploySW .apk or .ipa
DeployCustomGuitar
«protocol»
+SRTP
«protocol»
+SRTP
«device»
CustomGuitarServer
«database»
Oracle
«protocol»
+SRTP
«device»
Server
«database»
Oracle

Fig 10.1

Package Diagram

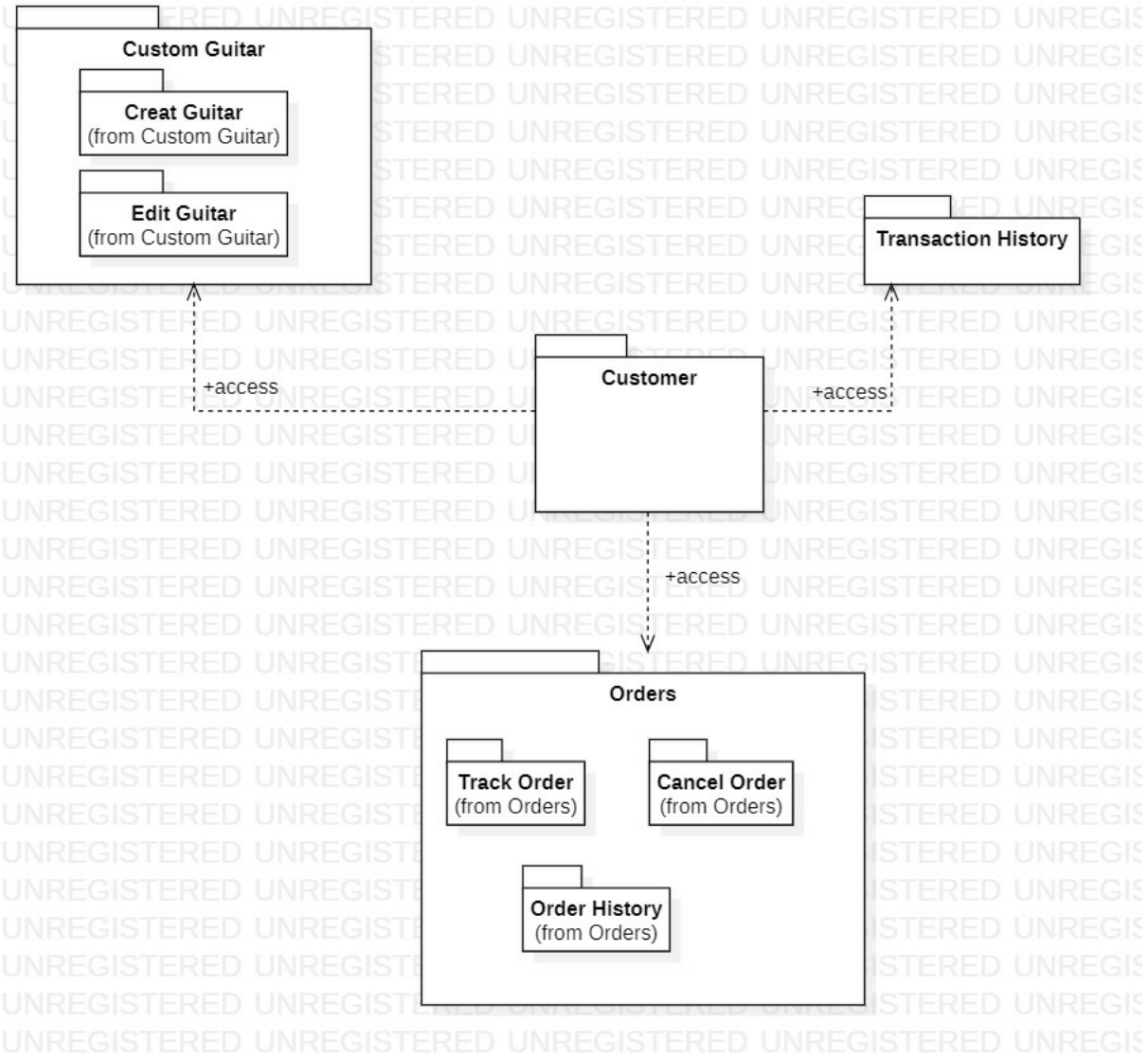

Fig 11.1

12. Conclusion

Guitars are unique musical instruments that are a joy to listen to, which is why in almost every band there is usually a guitarist.

To help guitarists in customising their own guitar according to their choice, we have made this custom guitar builder app which will ease the customising process.

13. References

- https://www.haloguitars.com/store/custom-guitars.html
- https://www.balaguerguitars.com/

Assignment Made by:

Shivang Bhatnagar (RA19119033010119) and Palak Sharma (RA1911033010112)