traveler

Project Title : Hotel Booking and Management (Traveler)
Group No. : 2
Group Members :

Washio Ferdous Rubai	(ID: 2231107642)
Nabila Tabassum Oyshi	(ID: 2231386042)
Mahmuda Zannat Punny	(ID: 2231518642)
Wasif Haider	(ID: 2232829642)

List of packages and classes:

AbsoluteMain		
	MainDev	(Washio Ferdous)
	MainUser	(Washio Ferdous)
Common		
	AboutProject	(Nabila Tabassum Oyshi)
	ArrayList	(Washio Ferdous)
	CPanel	(Nabila Tabassum Oyshi)
	CustomException	(Washio Ferdous)
	DevInfo	(Mahmuda Zannat Punny)
	FileReader	(Washio Ferdous && Nabila Tabassum Oyshi)
	FrameDev	(Wasif Haider)
	FrameUser	(Wasif Haider)
	HotelInfoButton	(Washio Ferdous)
	HotelInfoPanel	(Washio Ferdous)
	Hotels	(Mahmuda Zannat punny)
	IconAdder	(Washio Ferdous)
	Info	(Washio Ferdous)
	ObjecSaver	(Washio Ferdous)
	PointerScrollPanel	(Wasif Haider)
	ProfilePanel	(Washio Ferdous)
	ProfileScrollPanel	(Washio Ferdous)
	ResaveDev	(Nabila Tabassum Oyshi)
	ResaveUser	(Washio Ferdous)
	ThreadLogin	(Washio Ferdous)
	TravelPanel	(Wasif Haider)
	UserInfo	(Mahmuda Jannat Punny)
DevMain		
	DevLogInPanel	(Washio Ferdous)
	DevUIGenerator	(Washio Ferdous)
	HotelTable	(Mahmuda Zannat Punny)
	InfoPanel	(Washio Ferdous)
	MapPanel	(Washio Ferdous)
	MapView	(Washio Ferdous)
	StaticDev	(Nabila Tabassum Oyshi)
	UserTable	(Washio Ferdous && Mahmuda Zannat Punny)

- UserMain
  - MapImage (Washio Ferdous)
  - MapMain (Washio Ferdous)
  - SettingPanel (Nabila Tabassum Oyshi)
  - StaticUser (Nabila Tabassum Oyshi)
  - TravelHistory (Wasif Haider)
  - UserLogPanel (Nabila Tabassum Oyshi)
  - UserUIGenerator (Washio Ferdous)

Introduction

This Java program is a hotel booking and management system. It has two interfaces, one for the user and another for the developer. Each interface has its own unique functionalities and features.

Interface for the users:
Multiple users can have their own profiles and log in to their own profiles using their credentials. The users can do the following:

- View the hotels in a specific city in a list format and also in a map view.
- Check the pricing, availability, ratings, and contact details of the hotels.
- Book any hotel based on the number of adults and children that will stay and the number of days they will be staying.
- Get a receipt with their total bill and their own information.
- Each user can view their individual travel history, which is a list of hotels they have booked in the past and the details of those bookings.
- Change their avatar and map pointer design.
- Update their information and password.

Interface for the developers:
Multiple developers can have their own profiles and log in to their own profiles using their credentials. Developers have 3 levels of access, 1) CEO, 2) Admin, 3) Editor. There can only be a single CEO, and there can be multiple Admins and Editors. The avatars of the developers will vary depending on their access levels. The developers can do the following:

- Add hotels to any city of their choosing, with the details of the hotel such as the name of the hotel, the exact position of the hotel on the map, set ratings, contact information, and pricing for the hotel.
- View the list of hotels along with their information sorted by city.
- View the list of users and the list of developers sorted by their access level.
- Add new employees to the developer team.
- Update their passwords.

UML Diagram

<<CPanel>>
Interface

+ panel() : JPanel

AboutProject
Extends JPanel
Implements CPanel

- lPie : JLabel
- l9 : JLabel
- l25 : JLabel
- l60 : JLabel
- l6 : JLabel
- l1 : JLabel
- l2 : JLabel
- l3 : JLabel
- l4 : JLabel
- id1 : JLabel
- id2 : JLabel
- id3 : JLabel
- id4 : JLabel
- m1 : JLabel
- m2 : JLabel
- m3 : JLabel
- m4 : JLabel
- b1 : JButton
- b2 : JButton
- l : JLable[]

# paintComponent(Graphics g)
+ panel() : JPanel
+ setup() : void

Info

+ name : String
+ mail : String
+ phone : String
+ nid : String

+ Constructors

HotelInfoButton
Extends JButton
Implements CPanel

- popPanel : JPanel
- btnPanel : JPanel
- hml : JLabel
- hrl : JLabel
- hpl : JLabel
- pin : ImageIcon[]
- ii : ImageIcon

+ Parameterised Constructor

ArrayList

+ cityName : String[]
+ month : String[]
+ geoImage : ImageIcon[]
+ darkGeoImage : ImageIcon[]
+ profile : ImageIcon[]
+ mapPin : ImageIcon[]

FileReader

- hotMap : HashMap<String, LinkedList<Hotels>>
- travelInfo : HashMap<String, LinkedList<String[]>>
- devMap : HashMap<String, DevInfo>
- userMap : HashMap<String, UserInfo>
- object : LinkedList<Object>

+ hotelHashMap() : HashMap<String, LinkedList<Hotels>>
+ travelHashMap() : HashMap<String, LinkedList<String[]>>
+ developerHashMap() : HashMap<String, DevInfo>
+ userHashMap() : HashMap<String, UserInfo>
+ objectList() : LinkedList<Object>

UserInfo
Extends Info

+ pass : String
+ status : String
+ imageIndex : int

+ Constructors

DevInfo
Extends Info

+ pass : String
+ position : String
+ id : String
+ imageIndex : int

+ Constructors

FrameUser
Extends JFrame

- bUpdate : JButton
- done1 : JButton
- done2 : JButton
- bInfo : JButton
- bPass : JButton
- pInfo : JPanel
- pUpdate : JPanel

HotelInfoPanel
Extends JPanel
Implements CPanel

- cityName : String
- hotName : String
- phn : String
- p : double
- color : Color
- container : JPanel
- starPanel : JPanel
- lBook : JLabel
- lHotel : JLabel
- lCity : JLabel
- price : JLabel
- phone : JLabel
- star : JLabel
- s5 : JLabel
- s4 : JLabel
- s3 : JLabel
- s2 : JLabel
- s1 : JLabel
- p1 : JPanel
- p2 : JPanel
- p3 : JPanel
- p4 : JPanel
- p5 : JPanel
- hight : int
- yShift : int
- btnh : int
- track : int
- r : int[]
- avg : double
- icon : ImageIcon

+ Parameterised Constructor
# paintComponent(Graphics g)
+ panel() : JPanel
+ resize() : void
+ randomColor() : Color

ResaveDev

+ ResaveDev(uInfo : HashMap<String, DevInfo>)

ResaveUser

+ ResaveUser(uInfo : HashMap<String, UserInfo>)

ThreadLogin
Implements Runnable, CPanel

- x : int
- p : JPanel
- s : int
- ii : int
- c : Color
- a : int

+ ThreadLogin()
+ panel() : JPanel
+ run() : void
+ getRandomColor() : Color

- pOption : JPanel
- pPass : JPanel
- f : JFrame
- bExit : JButton
- bBack : JButton
- user : UserInfo
- mail : JButton
- phone : JButton
- nid : JButton
- l1 : JButton
- l2 : JButton
- l3 : JButton
- pp : JPanel
- lName : JLabel
- lMail : JLabel
- lPhn : JLabel
- lNID : JLabel
- lPass : JLabel
- lCPass : JLabel
- tfName : JTextField
- tfMail : JTextField
- tfPhn : JTextField
- tfNID : JTextField
- tfPass : JPasswordField
- tfCPass : JPasswordField
- uInfo : HashMap<String, UserInfo>
- o : LinkedList<Object>

+ FrameUser()
+ saveInfo() : void
+ setUp() : void

PointScrollPanel
Implements CPanel

- selectPanel : JPanel
- mainPanel : JPanel
- spPanel : JPanel
- imageIndex : int
- currentProfile : JPanel
- save : JButton
- lTic : JLabel
- scrollPane : JScrollPane
- image : ImageIcon[]
- object : LinkedList<Object>

+ panel() : JPanel
- setSP(i : int) : JPanel
- setup() : void

ProfilePanel
Implements CPanel

- image : ImageIcon[]
- index : int
- profilePanel : JPanel
- bName : JButton

+ Parameterised Constructor
# paintComponent(Graphics g)
+ panel() : JPanel

DevUIGenerator
Extends JFrame

- pSB : JPanel
- pExit : JPanel
- UIPanel : JPanel
- pExit2 : JPanel
- lExit : JButton
- fExit : JFrame
- bExit : JButton
- logBtn : JButton
- logPanel : DevLogInPanel
- mapView : MapView
- option : JButton
- home : JButton
- userDev : JButton
- hotelInfo : JButton
- logOut : JButton
- dev : DevInfo
- profile : ProfilePanel
- i1 : ImageIcon
- i2 : ImageIcon
- i3 : ImageIcon
- i4 : ImageIcon
- i5 : ImageIcon
- i6 : ImageIcon
- l1 : JLabel
- l2 : JLabel
- l3 : JLabel
- l4 : JLabel
- l5 : JLabel
- l6 : JLabel

+ DevUIGenerator(s : String)
+ DevUIGenerator()
- swap(i : int) : void
+ rmv(i : int) : void
- SetUp() : void

HotelTable
Implements CPanel

- cityName : JComboBox<String>
- hotelList : HashMap<String, LinkedList<Hotels>>
- vector : Vector<String>
- pHotTable : JPanel
- logo : JPanel
- columnNames : String[]
- option : JPopupMenu
- remove : JMenuItem
- update : JMenuItem
- model : DefaultTableModel
- table : JTable
- tableHeader : JTableHeader
- tfName : JTextField
- tfPhone : JTextField
- tfX : JTextField
- tfY : JTextField
- tfPrice : JTextField
- lName : JLabel
- lPhone : JLabel

InfoPanel
Implements CPanel

- xLoc : int
- yLoc : int
- pInfo : JPanel
- lx : JLabel
- ly : JLabel
- ltf : JLabel
- lavg : JLabel
- l1 : JLabel
- l2 : JLabel
- l3 : JLabel
- l4 : JLabel
- l5 : JLabel
- l6 : JLabel
- l7 : JLabel
- l8 : JLabel
- random : JButton
- add : JButton
- set : JButton
- tf1 : JTextField
- tfx : JTextField
- tfy : JTextField
- tfp : JTextField
- s5 : JTextField
- s4 : JTextField
- s3 : JTextField
- s2 : JTextField
- s1 : JTextField
- tfphn : JTextField
- pPoint : JPanel
- bdl : JLabel
- cb : JComboBox<String>
- option : JPopupMenu
- popRand : JMenuItem
- popLoc : JMenuItem

+ panel() : JPanel
-addPoint(x : int, y : int) : void
- allHotels(city : String) : void
-setRand() : void

PointerScrollPanel
Implements CPanel

- selectPanel : JPanel
- spPanel : JPanel
- imageIndex : int
- currentProfile : JPanel
- save : JButton
- lTic : JLabel
- scrollPane : JScrollPane
- mainPanel : JPanel
- images : ImageIcon[]
- object : LinkedList<Object>

+ panel() : JPanel
- setSP(i : int) : JPanel
- setUp() : void

```
- lX : JLabel
- lY : JLabel
- lPrice : JLabel
- save : JButton
- i : int
```

```
+ panel() : JPanel
- updateSetUp(name : String , city : String , x : int , y : int, price : double, phone : String) : void
- remove( hotelName : String, cityName : String) : void
- update( name : String , city : String , x : int , y : int, price : double, phone : String) : void
- setTable() : void
- resetFile() : void
- randomColor() : Color
```

UserTable
Implements CPanel

```
- devMap : HashMap<String, LinkedList<DevInfo>>
- dhm : HashMap<String, DevInfo>
- uInfo : HashMap<String, UserInfo>
- mode : JComboBox<String>
- dev : JComboBox<String>
- position : JComboBox<String>
- state : JComboBox<String>
- userColumn : String[]
- devColumn : String[]
- pTableUser : JPanel
- pTableDev : JPanel
- pTableMain : JPanel
- logo : JPanel
- modelUser : DefaultTableModel
- modelDev : DefaultTableModel
- userTable : JTable
- devTable : JTable
- bAdd : JButton
- reset : JButton
- cancel : JButton
- dSave : JButton
- uSave : JButton
- option : JPopupMenu
- remove : JMenuItem
- update : JMenuItem
- tfName : JTextField
- tfMail : JTextField
- tfPhone : JTextField
- tfId : JTextField
- tfNid : JTextField
- tfUPass : JTextField
- tfUMail : JTextField
- tfUphone : JTextField
- lName : JLabel
- lPosition : JLabel
- lMail : JLabel
- lPhone : JLabel
- lUMail : JLabel
- lUPhone : JLabel
- lUPass : JLabel
```

ProfileScrollPanel
Implements CPanel

```
- selectPanel : JPanel
- spPanel : JPanel
- user : UserInfo
- imageIndex : int
- currentProfile : JPanel
- save : JButton
- lTic : JLabel
- scrollPane : JScrollPane
- mainPanel : JPanel
- uInfo : HashMap<String, UserInfo>
- images : ImageIcon[]
- object : LinkedList<Object>
```

```
+ panel() : JPanel
- setSP(i : int) : JPanel
- setUp() : void
```

UserLoginPanel
Implements CPanel

```
- mainPanel : JPanel
- loginPanel : JPanel
- createPanel : JPanel
- pConfirm : JPanel
- name : JTextField
- email : JTextField
- logName : JTextField
- phone : JTextField
- nid : JTextField
- logPass : JPasswordField
- pw1 : JPasswordField
- pw : JPasswordField
- blogin : JButton
- create : JButton
- b1 : JButton
- lError : JLabel
- lLogIn : JLabel
- lCreate : JLabel
- loginname : JLabel
- password : JLabel
- account : JLabel
- createname : JLabel
- createmail : JLabel
- createphone : JLabel
- createnid : JLabel
- createPass : JLabel
- createNewPass : JLabel
- logBack : JLabel
- isLoged : boolean
- logerID : UserInfo
- object : LinkedList<Object>
```

```
+ panel() : JPanel
- log( name : String, pass : String) : void
- randomColor() : Color
```

- lUState : JLabel
- lId : JLabel
- lNid : JLabel
- DTH : JTableHeader
- UTH : JTableHeader
- devInfo : DevInfo

+ panel() : JPanel
- resetAll() : void
- setUsetTable() : void
- setDevTable() : void
- setUp() : void
- randomColor() : Color

ProfileScrollPanel

Implements CPanel

- fExit : JFrame
- lExit : JButton
- bExit : JButton
- option : JButton
- home : JButton
- settings : JButton
- travel : JButton
- logBack : JButton
- log : JButton
- pSB : JPanel
- mainPanel : JPanel
- pExit : JPanel
- pExit2 : JPanel
- mapView : MapMain
- i1 : ImageIcon
- i2 : ImageIcon
- i3 : ImageIcon
- i4 : ImageIcon
- i5 : ImageIcon
- i6 : ImageIcon
- i7 : ImageIcon
- l1 : JLabel
- l2 : JLabel
- l3 : JLabel
- l4 : JLabel
- l5 : JLabel
- l6 : JLabel
- profile : ProfilePanel

+ UserUIGenerator(s : String)
- swap(i : int) : void
- setUp() : void
- randomColor() : Color

HotelInfoButton

Implements CPanel

- travelMain : JPanel
- spPanel : JPanel
- uInfo : UserInfo
- travel : HashMap<String, LinkedList<String[]>>

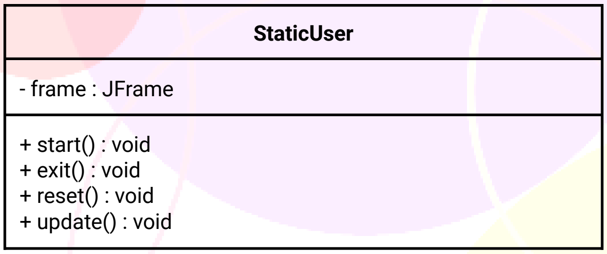
+ panel() : JPanel

MapMain

Implements CPanel

- hashHotel : HashMap<String, LinkedList<Hotels>>
- fDay : JComboBox<String>
- fMonth : JComboBox<String>
- fYear : JComboBox<String>
- tDay : JComboBox<String>
- tMonth : JComboBox<String>
- tYear : JComboBox<String>
- object : LinkedList<Object>
- cities : String[]
- tfSearch : JPanel
- buttonPop : JPanel
- buttonSearch : JPanel
- backPanel : JPanel
- mapPanel : JPanel
- pBook : JPanel
- pBookTime : JPanel
- pConfirm : JPanel
- logo : JPanel- pMap : JPanel
- pointPanel : JPanel
- spPanel : JPanel
- lError : JLabel
- lName : JLabel
- lMail : JLabel
- lPhone : JLabel
- lNid : JLabel
- lFrom : JLabel
- lTo : JLabel
- lAdult : JLabel
- lChild : JLabel
- fHotel : JLabel
- fCity : JLabel
- fDate : JLabel
- fTotalDay : JLabel
- fRoom : JLabel
- fPrice : JLabel
- tfName : JTextField
- tfMail : JTextField
- tfPhone : JTextField
- tfNid : JTextField
- tfAdult : JTextField
- tfChild : JTextField
- bBook : JButton
- bCont : JButton
- bLogIN : JButton
- cancel1 : JButton
- cancel2 : JButton
- cancel3 : JButton
- popSearch : JPopupMenu
- bookHotel : String
- bookCity : String
- bookPrice : double

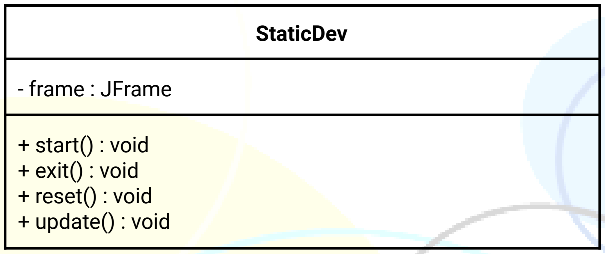
+ panel() : JPanel
- allHotelPanel(s : String) : void
- allHotels(city: String) : void
- setUp() : void
- randomColor() : Color



StaticUser

- frame : JFrame

+ start() : void
+ exit() : void
+ reset() : void
+ update() : void



StaticDev

- frame : JFrame

+ start() : void
+ exit() : void
+ reset() : void
+ update() : void

Description

AbsoluteMain :

- **MainDev :** One of the two main classes which creates a static developer JFrame.
- **MainUser :** One of the two main classes which creates a static user JFrame.

Common :

- **AboutProject :** This class is used to display the information in the about section of the program. *[Extends : JPanel, Implements : CPanel]*
- **ArrayList :** Stores some commonly used arrays that are all public so that it can be accessed from any class.
- **CPanel :** Only interface class that holds a method which returns a JPanel and has been implemented by all those classes that creates a panel.
- **CustomException :** One custom exception class that can handle all the self-defined exceptions. *[Extends : Exception]*
- **DevInfo :** This class is used to store the info of the currently logged-in developer. *[Extends : Info]*
- **FileReader :** This class holds methods that can read files and return the HashMap or LinkedList of that file's information.
- **FrameDev :** This class is used to display the information of the currently logged-in developer. The password of the developer can also be updated using this class. *[Extends : JFrame]*
- **FrameUser :** This class is used to display the information of the currently logged-in user. The password as well as the information of the user can also be updated using this class. *[Extends : JFramel]*
- **HotelInfoButton :** This class is used to show the info of the hotel like the hotel name, ratings, pricing, and location on the card. *[Extends : JButton, Implements : CPanel]*
- **HotelInfoPanel :** This processes and creates a JPanel that stores all the information of a hotel on a panel which is resizable. *[Extends : JPanel, Implements : CPanel]*
- **Hotels :** This class is used to store the information of the hotel.
- **IconAdder :** This class is used to add more than one ImageIcon in one JLabel.
- **Info :** This is the parent class of DevInfo and UserInfo class.
- **ObjectSaver :** "object.txt" is a special type of file that is used for storing login information and user setting information.

- **PointerScrollPanel :** This class is used to display the scrolling panel to change the pointer icon in the user's settings page.*[Implements : CPanel]*
- **ProfilePanel :** This class is used to show the profile image and name of the user/developer on the sidebar and is also used to figure out which panel will be opened once the profile picture is clicked(FrameDev/FrameUser). *[Extends : JPanel, Implements : CPanel]*
- **ProfileScrollPanel :** This class is used to display the scrolling panel to change the profile picture in the user's settings page. *[Implements : CPanel]*
- **ResaveDev :** This class is used to save the developer's info after any changes to it.
- **ResaveUser :** This class is used to save the user info after any changes to it.
- **ThreadLogin :** This class is used to show the bubbles animation on the login page using threading. *[Implements : Runnable, CPanel]*
- **TravelPanel :** This class is used to display the individual listings of where the user has traveled to, in the travel history page. *[Implements : CPanel]*
- **UserInfo :** This class is used to store the info of the currently logged-in user. *[Implements : Info]*

DevMain :

- **DevLoginPanel :** This class is used to show the login page of the developer window. *[Implements : CPanel]*
- **DevUIGenerator :** This class generates the main frame for developers that has been displayed but this class doesn't have the power to show or dispose of the frame. *[Extends : JFrame]*
- **HotelTable :** This class is used to show the list of all hotels including their info in the developer window. *[Implements : CPanel]*
- **InfoPanel :** This is the first window that opens on the developer frame which stores all the components to add a new hotel on the map with all the basic information of the existing hotels. *[Implements : CPanel]*
- **MapPanel :** This class is used to display the map image in the developer window. *[Extends : JPanel, Implements : CPanel]*
- **MapView :** This class processes the map images that have been created in MapPanel and returns a JPanel. *[Implements : CPanel]*
- **StaticDev :** This creates a static JFrame of DevUIGenerator class and can statically display, dispose, restart or update the frame.
- **UserTable :** This class is used to show the list of users and developers in the developer window. *[Implements : CPanel]*

UserMain :

- **MapImage :** This class is used to display the map image in the user window. *[Extends : JPanel, Implements : CPanel]*
- **MapMain :** The first window that opens on the user frame and contains all the components to view all the hotel information and to book a hotel room. *[Implements : CPanel]*

- **SettingPanel :** This class displays the user settings page in the user window. *[Extends : JPanel, Implements : CPanel]*
- **StaticUser :** This creates a static JFrame of UserUIGenerator class and can statically display, dispose, restart or update the frame.
- **TravelHistory :** This class is used to display the travel history page of the currently logged-in user. *[Implements : CPanel]*
- **UserLogPanel :** This class is used to show the login page of the user window. *[Implements : CPanel]*
- **UserUIGenerator :** This class generates the main frame for developers that has been displayed but this class doesn't have the power to show or dispose of the frame. *[Extends : JPanel]*

Files :

- **devInfo.txt :** Stores developers information.
- **userInfo.txt :** Stores users information.
- **hotels.txt :** Stores hotel information.
- **travelInfo.txt :** Stores all the traveling records and histories of all users.
- **info.txt :** Stores the project information that has been scanned in the AboutProject class.
- **object.txt :** Stores spatial type of information to keep track of the user login and setting changes.

Swing Description

In our project, we made things look good by using Java Swing. We split our work into two parts. Each window had sections for different things we wanted to show. To make it all look neat and organized, we used panels inside the windows. With Swing, we could make buttons and stuff look how we wanted and made sure everything worked smoothly when people clicked on things. It helped us make a good-looking and easy-to-use program.

Apart from Swing, we also added pictures and designs to our project to make it more interesting. We used Java's graphics tools and algorithm to create random designs in the background. These designs weren't just for looks – they also helped show things in a cool way. By using both Swing and graphics, our project not only works well but also looks nice and fun.

Discussion

During our project, we faced challenges like learning graphics, switching images with buttons, and implementing a combo box to change the interface. To tackle these, we used special techniques. We learned graphics by checking online guides and tutorials. Learning about switching images with buttons took a lot of our project time. By researching, trying things out, and working together, we found solutions for each problem. This helped us create a project that works well despite the challenges we faced.

Future Plans

This is a complete project without any bugs and contains everything that we wanted it to have. Still nothing is perfect and we can improve it in many ways like adding changeable theme settings and pdf recites. But we don't have any future plans regarding this project as this is not an unique idea and not a practical project in a sense that we can develop an app or a website without the use of this base level GUI library and still it will be both easier to build for the developer and visually more appealing to the user. For that reason we also don’t have any future plans to launch this yet.

Conclusion

From this project, we learned how to work together in a group, how to implement Java Swing and graphics, and most importantly, to run this project successfully. We learned how to solve problems and how to push ourselves to tackle any challenges. Projects are never complete with existing knowledge. This is a process of learning new things and practically implementing all those things that we know.