IMPLEMENTATION DOCUMENT

Software Engineering

Supervisor:
Prof. Matteo Giovanni Rossi

Authors:	ID:
Ahmed Ibrahim Abdelrazzak Hamed	10682755
Khaled Said Ahmed Maamoun	10696857
Mahmoud Mohamed Aboelwafa Medany	10715340

Table of Contents

1-INTRODUCTION

Overview

This document is a follow up on the previous two, "Requirement Analysis and Specification Document" and "Design Document"; it will describe the prototype for the website and how you can build the website and how to modify by describing how each library and component work.

2- TOOLS, FRAMEWORKS, AND FUNCTIONALITIES

Tools

The website has been built based on this tools.

- Visual Studio IDE
- ReactJS JavaScript library
- React Redux
- Redux thunk
- Redux Firebase
- Axios (Promised-based HTTP client for JavaScript)
- Maps JavaScript API
- Geocoding API
- Places API
- Google Firebase API

Framework

The website has been built as a cross-platform website application using ReactJS and jQuery framework. The programming language used is JavaScript. With React JS you can build a single page application which is way faster and smoother to navigate inside.
Likewise, it was possible to navigate in different web application components inside one HTML page instead of requesting the server to load a new HTML page each time for every single component, which made the transition responsive and quicker.

As said in the Design document, the client-server model is achieved with Google Firebase with the main player of our data synchronization functionality is the cloud Firestore. Personalized experience as well as user data security (both customer and AYLI) is managed through Firebase Authentication backend services. While notification functionality is implement using Firebase Cloud Functions, which is a serverless framework that lets automatically run backend code in response to events triggered by Firebase features. All the pictures uploaded by the AYLIs or users is managed by Firebase storage, an object storage service that is powerful, simple, and cost-effective. Whereas for storing all other data, including service details, proposal details, messages, etc., we used Firebase Cloud Firestore which is a real-time database. Moreover, after everything ready, we deployed the application using Firebase hosting.

The user interface is developed using ReactJS, a declarative, efficient, and flexible JavaScript library for building user interfaces. With this, we develop complex user interface from small and isolated pieces of code called “components”. Other huge advantage from ReactJS is the possibility to use reusable user interface component and implement single page application.

Functionalities

Referring to the Requirements and the Design documents, we have implemented most of the functionalities stated. First of all, the customer must sign into his account in order to access to the home page of the website.

Customer page

The main interface of AYLI consists of choosing which type of AYLI you would like to search for, the options are Hotel, Hostel, Restaurant and Entertainment.

- Customer can use filter feature to search for Hotel. The filter included are filter by sea view, average prices, ratings, single, couples, families classic, modern, nearby, and ratings.
- Customer can use filter feature to search for Hostels, Restaurant, Entertainment. The filter included are filter by Opening hours, average prices, ratings, single, couples, families classic, modern, nearby, and ratings.

A map that allows the customer to search for an AYLI around. When a customer chooses an AYLI from the map a request will be send to the AYLI itself to send a proposal for the customer.

The customer can add any extra notes or comment in the custom specification section, in case he has any specific preferences or requests.

3-STRUCTURE OF THE SOURCE CODE

Firebase database

The connection between Google Firebase API and the website as well as their interaction are managed using the following directories:

Store

/actions

- authActions.js

This action dispatcher is the authentication manager. It is responsible of verifying credential of the user by performing a query in the Firebase authentication service that enables them to perform sign in and sign out functionality.
It is also responsible to perform sign up functionality using Firebase cloud Fire-store.

- messageActions.js

This action dispatcher is the messaging manager. It allows customer and an AYLI to communicate by creating messages and share together.

- postActions.js

This action dispatcher is the broadcast manager. customer and an AYLI to post in the broadcast.

- proposalActions.js

This action dispatcher is the proposal, invoice, and automatic delivery message manager. It allows any type on AYLI to create a proposal and it changes the status of the invoice. Whereas for the Customer, it allows them to rate the experience, file claim for an AYLI, and to pay the invoice.
It is also allowing the Customer to confirm or reject a proposal sent by an AYLI and generate invoice for both. This action dispatcher also allows the system to inform the Customer that the payment is received. Finally, it allows the website to automatically recommend about another AYLI to manage the customer request

- serviceActions.js

This action dispatcher is the service manager. It enables the customer to pick an ayli from the map and choose it, whereas for the AYLI It enables it enables them to confirm, reject, or recommend.

/reducer

- authReducer.js

It is responsible of evaluating the action dispatched by the authentication manager.

- messageReducer.js

It is responsible of evaluating the action dispatched by the messaging manager.

- postReducer.js

It is responsible of evaluating the action dispatched by the broadcast manager.

- proposalReducer.js

It is responsible of evaluating the action dispatched by the proposal, invoice, and automatic delivery message manager.

- serviceReducer.js

It is responsible of evaluating the action dispatched by the service manager.

- rootReducer.js

It is the container of all the reducers of the application.

config

- **fbConfig.js**

It is configuration file to connect Google Firebase project to our application.

Functions

- **index.js**

It contains the cloud function which run on server side (not on client side). This function notifies the user according to the requirement.

Website Application

The interaction between the application layer, the database layer, and the presentation layer and in particular the execution of the application layer as well as the layout of the presentation layer are assured using the following directories:

auth

- SignIn.js

This React component contains the sign in form using the JSX template and allows the interaction between the customer and the form through some JavaScript code.

- SignUp.js

This React component contains the sign-up form using the JSX template and allows the interaction between the customer and the form through some JavaScript code.

BufferOfRejectedServices

- Buffers.js

This React component allows passing props to the BufferList.js child component using JSX template and some JavaScript code.

- BufferList.js

This React component is responsible of passing a list of the available requests in the buffer to BufferSummary.js child component using JSX template.

- BufferSummary.js

This React component allows to display the preview of the requests available in the buffer using JSX template.

- BufferDetails.js

This React component is responsible of displaying the details of the selected service by the AYLIs using the JSX template and of getting data from the Firebase Cloud Fire-store through some JavaScript code.

dashboard

- Contact.js

This React component allows the Customer, using a form, to contact the administrator or to file a claim a service conducted by an AYLI to the administrator using JSX template and JavaScript code to allow interaction between the Customer and the form.

- Dashboard.js

This React component is the homepage of the Customer account. It enables him to visualize the nearby AYLI on Google Maps using the Maps JavaScript API and apply a filter to the AYLIs list. It also allows the localization of the Customers and visualization of the localization in friendly format using Places API, Geocode API, and Axios. It enables also the customer to launch a request by clicking on the desired AYLI. All these features are using JSX template and JavaScript code to allow interaction between the customer and the form.

- Broadcast.js

This React component allows passing props to the PostList.js child component and also to the Notifications.js child component using JSX template and some JavaScript code.

- InBox.js

This React component allows, based on user type, either passing messages and receipts props in case of an AYLI or passing replied messages and AYLI receipts in case of a customer to the MessageList.js child component and also to the Receipts.js child component using JSX template and some JavaScript code.

- Notifications.js

This React component is responsible of displaying the list of the available notifications regarding adding a new post in the broadcast or regarding adding a new user to the platform using JSX template.

- InvoiceNotifications.js

This React component is responsible of displaying the list of the available notifications regarding the action performed on the invoices using JSX template.

- PropDecisions.js

This React component is responsible of displaying the list of the available notifications regarding the Customer proposal decision using JSX template.

- PropNotifications.js

This React component is responsible of displaying the list of the available notifications regarding new proposal received by the customer using JSX template.

- Proposals.js

This React component allows passing props to the ProposalList.js child component and also to the PropNotifications.js child component using JSX template and some JavaScript code

- Receipts.js

This React component is responsible of displaying the list of the available notifications regarding receiving a new message by the AYLI from customer using JSX template.

- ServiceCreatedNotif.js

This React component is responsible of displaying the list of the available notifications regarding creating a request by the customer using JSX template.

invoices

- InvoiceList.js

This React component is responsible of passing a list of the available invoices to the InvoiceSummary.js child component using JSX template.

- InvoiceSummary.js

This React component allows to display the preview of the invoices available belonging to the specific customer and AYLI using JSX template.

- InvoiceDetails.js

This React component is responsible of displaying the details of the selected invoice by the AYLI or Customer. If the user type is customer then the payment option is enabled as well as the rating option beside the invoice status is displayed as read only for the Customer where is this invoice status is displayed as read and write for the AYLI. These features are implemented using the JSX template and JavaScript Code to get data from the Firebase Cloud Fire-store and to ensure interactivity between the user and the input buttons.

- Payment.js

This React component is responsible of displaying a credit card registration form using JSX template and of controlling the form inputs and getting data from Firebase Cloud Fire-store using JavaScript code

layout

- Navbar.js

This React component is responsible of displaying the Navbar links of our application depending on the authentication status of the user and on the user type regarding the signed in links. These features are implemented using JSX template and JavaScript code to pass props to the SignedInLinks.js child component or to Signed Out Links, js child components and to get data from the File base Cloud Fire-store.

- SignedInLinks.js

This React component is responsible of displaying the signed in links in the Navbar of our application depending of the user type using JSX template and JavaScript code to get data from the File base Cloud Fire-store.

- SignedOutLinks.js

This React component is responsible of displaying the signed-out links in the Navbar of our application using JSC template.

messages

- MessageDetails.js

This React component is responsible of displaying the selected message details depending on the user type using JSX template and JavaScript code to get data from the File-base Cloud Fire-store.

- MessageList.js

This React component is responsible of passing a list of the available messages to the MessageSummary.js child component using JSX template.

- MessageSummary.js

This React component is responsible of displaying a preview of the list of available messages depending on the user type using JSX template.

- NewMessage.js

This React component is responsible of displaying a form of a new message for the customer using JSX template and of getting data from the Firebase Cloud Firestore as well as of adding and controlling the interactivity between the customer and the form inputs using JavaScript code.

- ReplyMessage.js

This React component is responsible of displaying a form of a new message for the AYLIs using JSX template and of getting data from the Firebase Cloud Fire-store.

posts

- CreatePost.js

This React component is responsible of displaying a form of new post for the AYLIs using JSX template and of getting data from the Firebase Cloud Fire-store.

- PostDetails.js

This React component is responsible of displaying the selected post details depending on the user type using JSX template and JavaScript code to get data from the Filebase Cloud Firestore.

- PostList.js

This React component is responsible of passing a list of the available posts to the PostSummary.js child component using JSX template.

- PostSummary.js

This React component is responsible of displaying a preview of the list of available posts depending on the user type using JSX template

proposals

- CreateProposal.js

This React component is responsible of displaying a form for the AYLIs to create a new proposal regarding a service received from the customer. The structure of this form is implemented using JSX template. In order to get data from the Firebase Cloud Firestore and in order to add and control the interactivity between the user and the form we use JavaScript code.

- Invoices.js

This React component is responsible of passing the invoices list as props to the InvoiceList.js child component as well as of passing the invoices notification list as props to the InvoiceNotification.js child component using JSX template and JavaScript code to get and manipulate the data from the Firebase cloud Fire-store.

- ProposalDetails.js

This React component is responsible of displaying the details of selected proposal by the Customer depending on the Customer decision regarding this proposal. This feature is implemented using JSX template and JavaScript to get and to control data from Firebase cloud Firestore.

- ProposalList.js

This React component is responsible of passing the list of available proposals for the ProposalSummary.js child component using JSX template.

- ProposalSummary.js

This React component is responsible of displaying the preview of each element of available proposals for current signed in Customer using JSX template.

services

- AllServiceList.js

This React component is responsible of passing the list of services for the current signed in an AYLI as props for the ServiceList.js child component as well as the corresponding notifications for the PropDecisions.js child component using JSX template and JavaScript to get and control the data from Firebase Cloud Fire-store.

- CreateService.js

This React component is responsible of displaying a form for the current signed in customers to create a request to a specific AYLI using JSX template and JavaScript code to control the interactivity between the user and the form and to get the data from the Firebase Cloud Fire-store.

- ServiceHistory.js

This React component is responsible of passing the list of services for the current signed in Customer as props for the ServiceList.js child component as well as the corresponding notifications for the RequestCreatedNotif.js child component using JSX template and JavaScript to get and control the data from Firebase Cloud Fire-store

- ServiceList.js.

This React component is responsible of passing the list of available services for the ServiceSummary.js child component using JSX template.

- ServiceRequestDetails.js

This React component is responsible of displaying the details of the service selected depending on the service decision using JSX template and JavaScript code to get and control data from Firebase Cloud Fire-store.

- ServiceSummary.js

This React component is responsible of displaying the preview of each service of the list using JSX template.

App.js

This is the root component where we can find all the parent and independent component.

index.css

This is the CSS file that is responsible on styling some parts of our User Interface.

index.js

This is the wrapper component of our root component App.js giving it enhancers of higher order that allow to link between React, Redux Thunk, Google Firebase API, and our root component.

index.html

This is the structure of our webpage, a single page application that encapsulate all of our components.

4-INSTALLATION INSTRUCTIONS

This software packages should be installed, to evaluate all the components of the website.

List of packages:

"antd"	"4.16.13"
"axios"	"0.22.0"
"body-parser"	"1.19.0"
"bootstrap"	"5.1.2"
"cookie-parser"	"1.4.5"
"core-util-is"	"1.0.3"
"cors"	"2.8.5"
"date-fns"	"2.25.0"
"emailjs"	"3.6.0"
"emailjs-com"	"3.2.0"
"emailjs-smtp-client"	"2.0.1"
"express"	"4.17.1"
"express-handlebars"	"5.3.4"
"firebase"	"9.1.1"
"google-maps-react"	"2.0.6"
"http-proxy-middleware"	"2.0.1"
"jquery"	"3.6.0"
"material-ui-time-picker"	"1.3.0"
"materialize-css"	"1.0.0"
"moment"	"2.29.1"
"nodemailer"	"6.6.5"
"nouislider"	"13.1.5"
"nouislider-react"	"3.4.1"
"react"	"17.0.2"
"react-credit-cards"	"0.8.3"
"react-dom"	"17.0.2"
"react-geocode"	"0.2.3"
"react-google-autocomplete"	"2.4.3"
"react-google-maps"	"9.4.5"
"react-gradient-timepicker"	"0.0.3"
"react-phone-input-2"	"2.14.0"
"react-phone-number-input"	"3.1.31"
"react-rating-stars-component"	"2.2.0"
"react-redux"	"7.2.5"
"react-redux-firebase"	"3.10.0"
"react-router-dom"	"5.3.0"
"react-scripts"	"4.0.3"
"react-time-picker"	"4.4.2"
"reactstrap"	"8.10.0"
"redux"	"4.1.1"
"redux-firestore"	"0.15.0"
"redux-thunk"	"2.3.0"
"web-vitals"	"1.1.2"

"wnumb"   "1.2.0"

The system components are the backend server and the website.

Back-end Server.

The backend server has been built using google Firebase services. In order to use google firebase you should follow those steps.

- Log in to your google account and connect to firebase with your account as an admin then from the console on the top right you can press on go to console.
- you will find list of projects depends on how many projects you have you should choose AYLI project to see the back-end server.
- After you choose the project you will find on the left a list which we will go through it now to explain it.

Authentication:

This service provides backend services, easy-to-use Software-development-kits, and ready-made User-Interface libraries to authenticate users for the sign-up process to the website.

Cloud Firestore:

This service provides flexible, scalable database for the website. From the Data option, we can Navigate through the collections that are used to store the data.Moreover, by selecting the rules option, we can see all the security rules for our website.

The used collections are:

- aux
- cards: contains all the registered credit cards inserted by the customers for payment
- claim notifications : contains all the confirming notifications upon claim creation created by the customers.
- claims: contains all the claims and the contact requests created by an AYLI
- companyPropDecisions: contains all the notifications corresponding to AYLIs regarding the Customer proposal decision
- companyReceipts: contains all the notifications corresponding tto an AYLI regarding the reception of customer messages
- confirmedServNotifications: contains all the notifications regarding action performed on services.
- Confirmed Proposals: contains all the confirmed proposals by the Customers.
- Confirmed Services: contains all the confirmed services by the AYLIs
- createInvNotifications: contains all the notifications corresponding to invoice generation.
- createServNotifications: contains all the notifications corresponding to new request created by the Customers.
- invoices: contains all the invoices generated.

- messages: contains all the messages sent by the Customers
- notifications: contains all the notifications regarding sign up of a new user and adding a new post.
- posts: contains all the post created by the AYLIs.
- propNotifications: contains all the notifications regarding the actions performed on proposals
- proposals: contains all the proposals created by the AYLIs.
- receipts: contains all the notifications regarding receiving a new message from the Customers
- rejectedProposals: contains all the rejected proposals by the Customers.
- rejectedServices: contains all the rejected services by the AYLIs and all the eliminated SOR.
- repMessages: contins all the messages created by AYLIs
- serviceStatus: contains and track the status of the service
- services: contains all the services created by the Customers
- users: contains all the signed up users.

Storage

This service provides powerful, simple, and cost-effective object storage service built for Google scale.

Hosting

Is to deploy the website to the firebase server

Functions

Visualize all the cloud function that we use for notifications

- claimLaunch: is a cloud function which is launched when a new claim is added and it invokes createNotifClaim to add the corresponding notifications.
- serviceConfirmed: is a cloud function which is launched when a new confirmed service is added and it invokes confirmServNotifications to add the corresponding notifications.
- invoiceCreated: is a cloud function which is launched when a new invoice is added and it invokes createInvNotifications to add the corresponding notifications.
- serviceCreated: is a cloud function which is launched when a new service is added and it invokes createServNotifications to add the corresponding notifications.
- proposalConfirmed: is a cloud function which is launched when a new confirmed proposal is added and it invokes createPropDecision to add the corresponding notifications.
- proposalRejected: is a cloud function which is launched when a new rejected proposal is added and it invokes createPropDecision to add the corresponding notifications.
- replyMessageCreated: is a cloud function which is launched when a new AYLI message is added and it invokes createReceiptCompany to add the corresponding notifications.

- Proposal-Created: is a cloud function which is launched when a new created proposal is added and it invokes create Prop Notification to add the corresponding notifications.
- Message-Created: is a cloud function which is launched when a new created message is added and it invokes create Receipt to add the corresponding notifications.
- Post-Created: is a cloud function which is launched when a new post is added and it invokes create Notification to add the corresponding notifications.
- User-joined: is a cloud function which is launched when a new user is added and it invokes create Notification to add the corresponding notifications.

Website application

1. The website have been built using Visual Studio Code IDE. In order to run website application in local server, operator should follow the steps below:
    1. Open Visual Studio code or any other IDE
    2. Import the project file.
    3. Open the terminal and write:

       cd AYLI

       npm start

    4. You will be automatically forwarded to the local server (http://localhost:3000)

   In order to open our website application, you can visit the following URL:

   https://ayli-fc138.web.app/signin

5-PICTURES OF THE WEBSITE

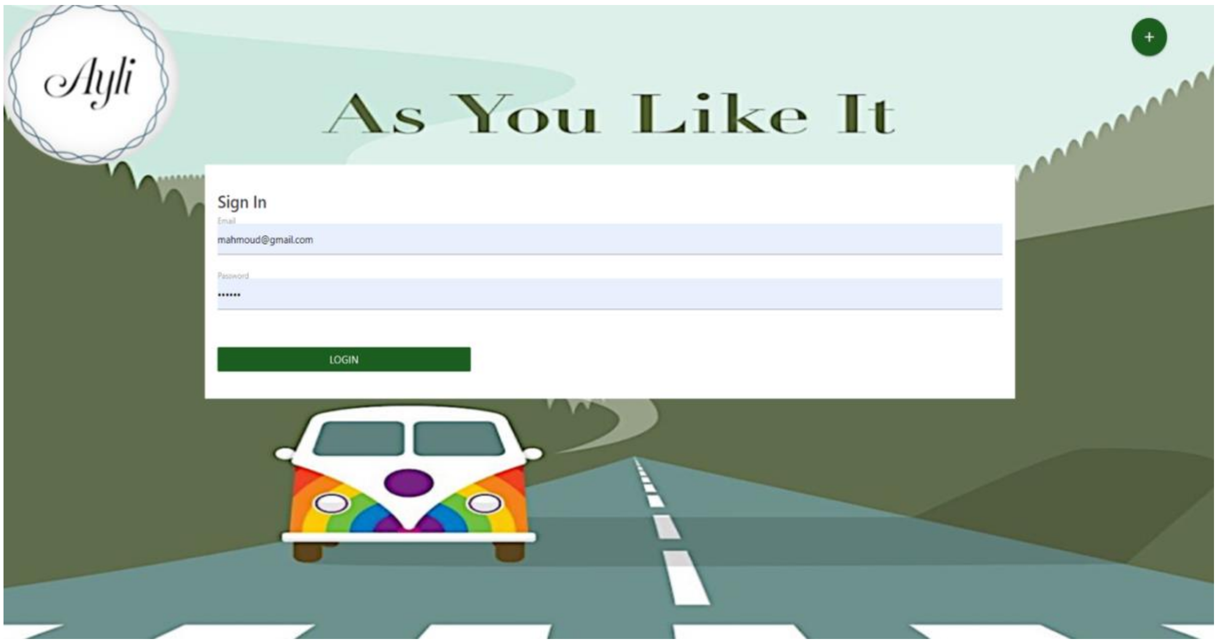


Figure 1 Sign In

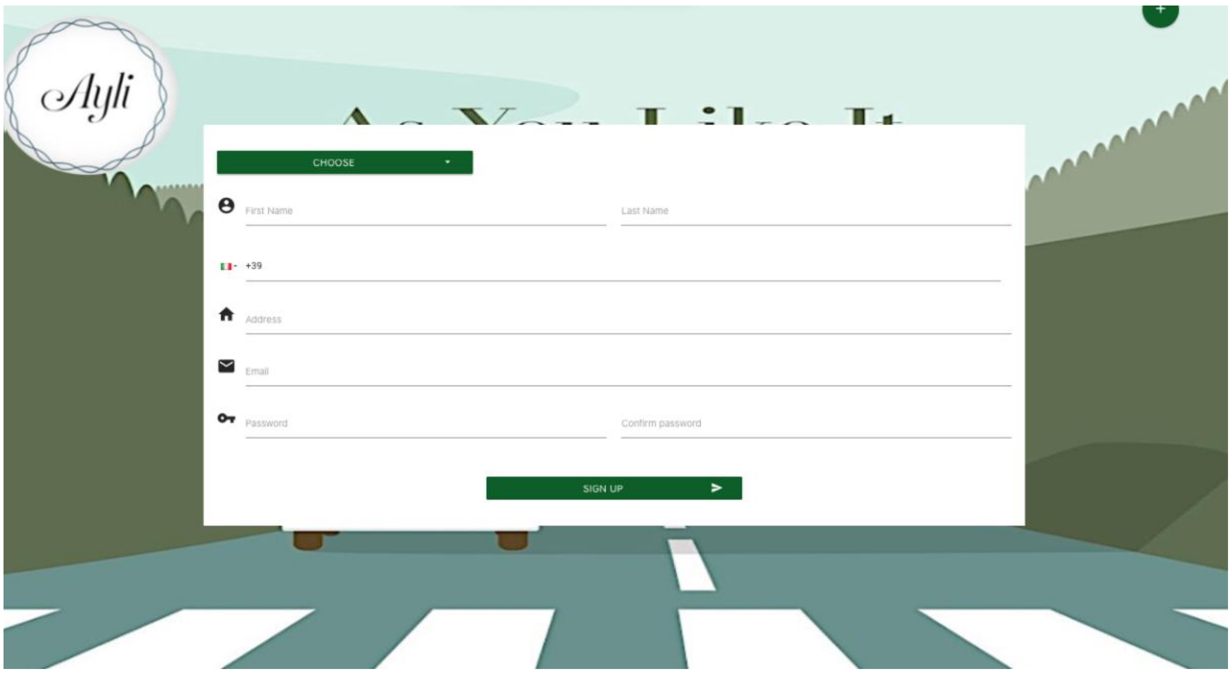


Figure 2 Customer signup

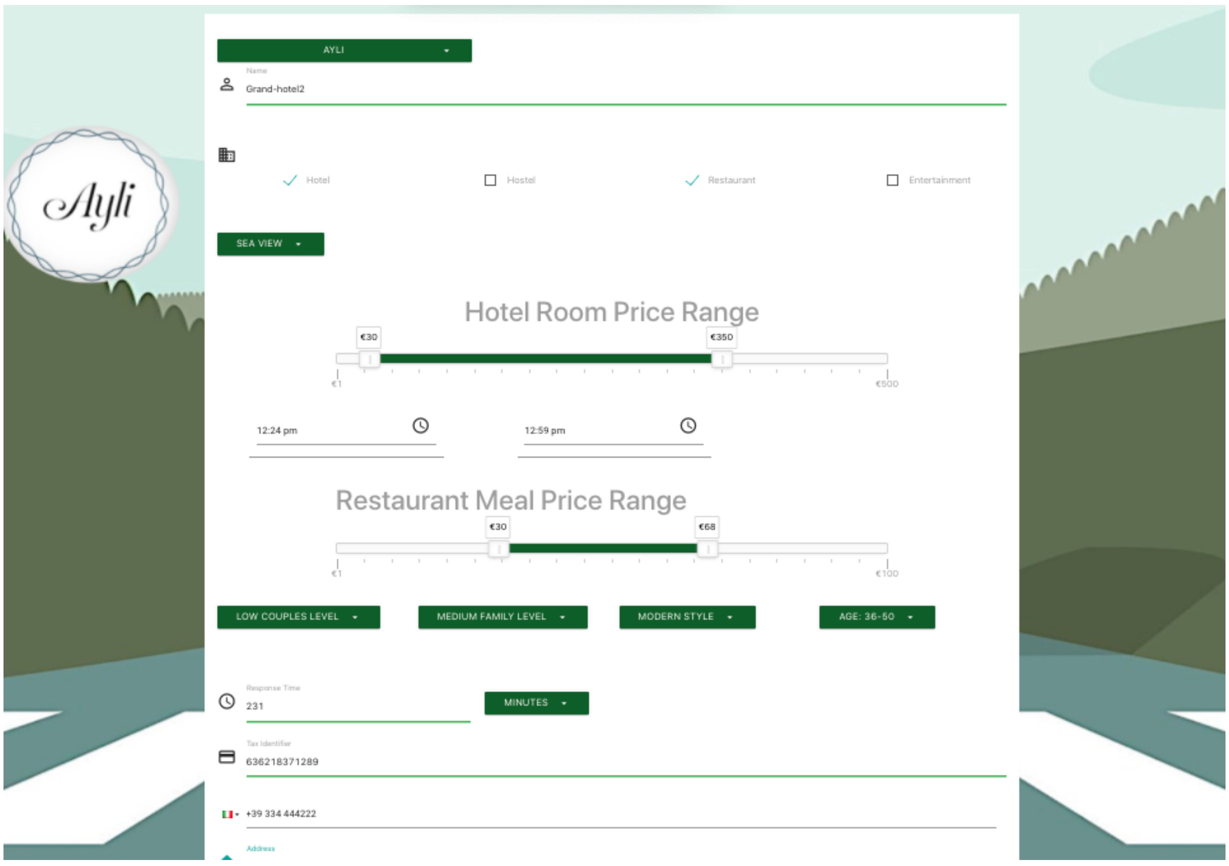


Figure 3 AYLI Sign up

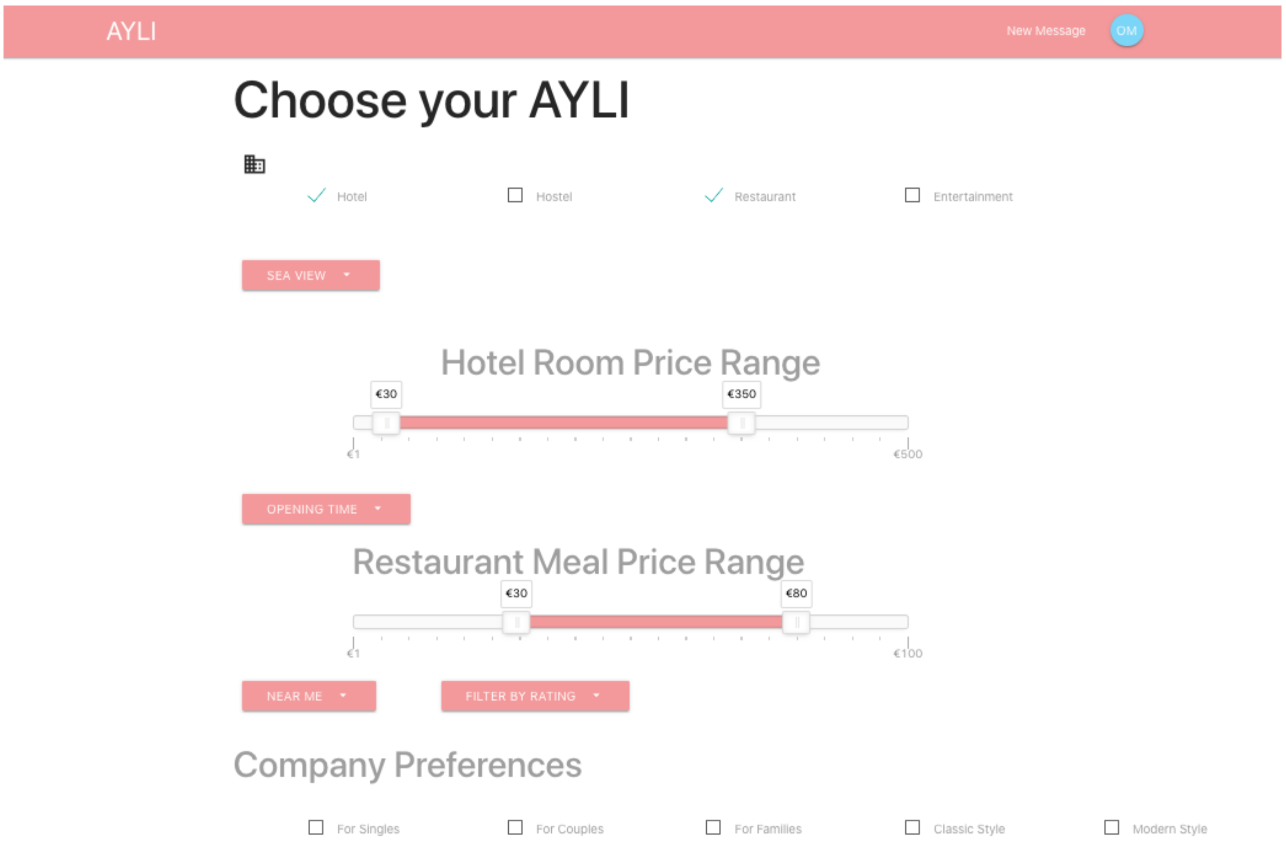
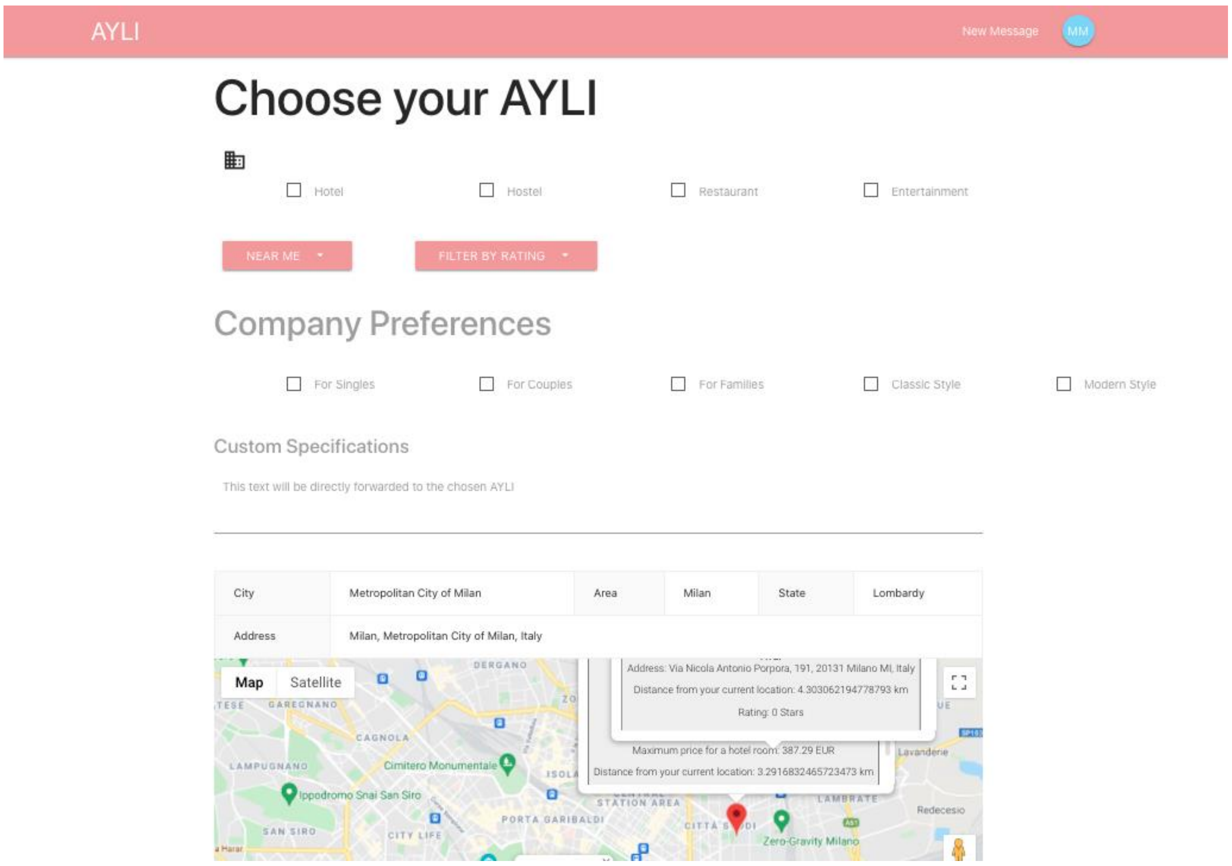


Figure 4 Customer Logged in



Choose your AYLI

- [ ] Hotel
- [ ] Hostel
- [ ] Restaurant
- [ ] Entertainment

Company Preferences

- [ ] For Singles
- [ ] For Couples
- [ ] For Families
- [ ] Classic Style
- [ ] Modern Style

Custom Specifications

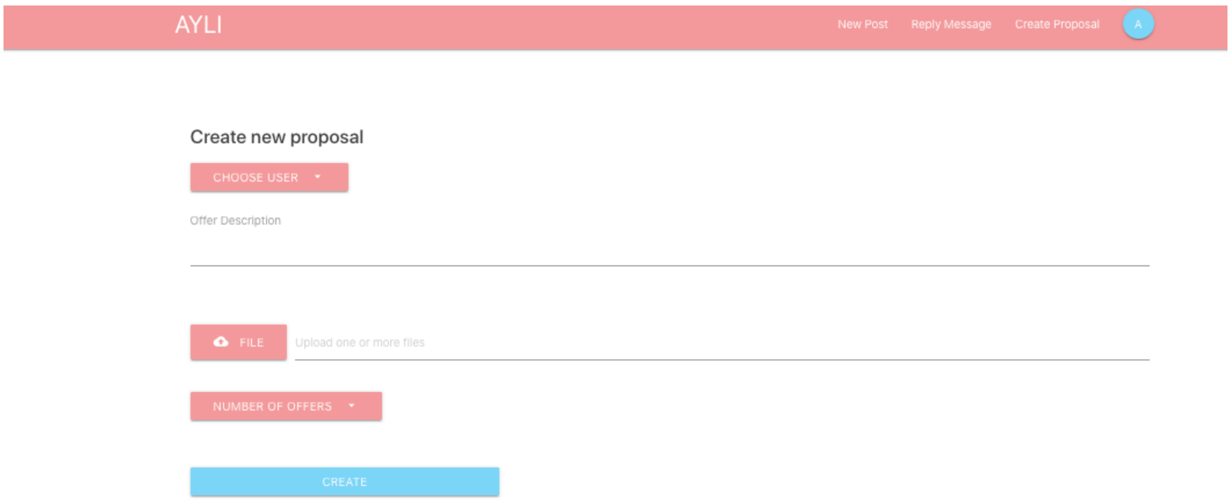
This text will be directly forwarded to the chosen AYLI

City	Metropolitan City of Milan	Area	Milan	State	Lombardy
Address	Milan, Metropolitan City of Milan, Italy				

Address: Via Nicola Antonio Porpora, 191, 20131 Milano MI, Italy
Distance from your current location: 4.303062194778793 km
Rating: 0 Stars

Maximum price for a hotel room: 387.29 EUR
Distance from your current location: 3.2916832465723473 km

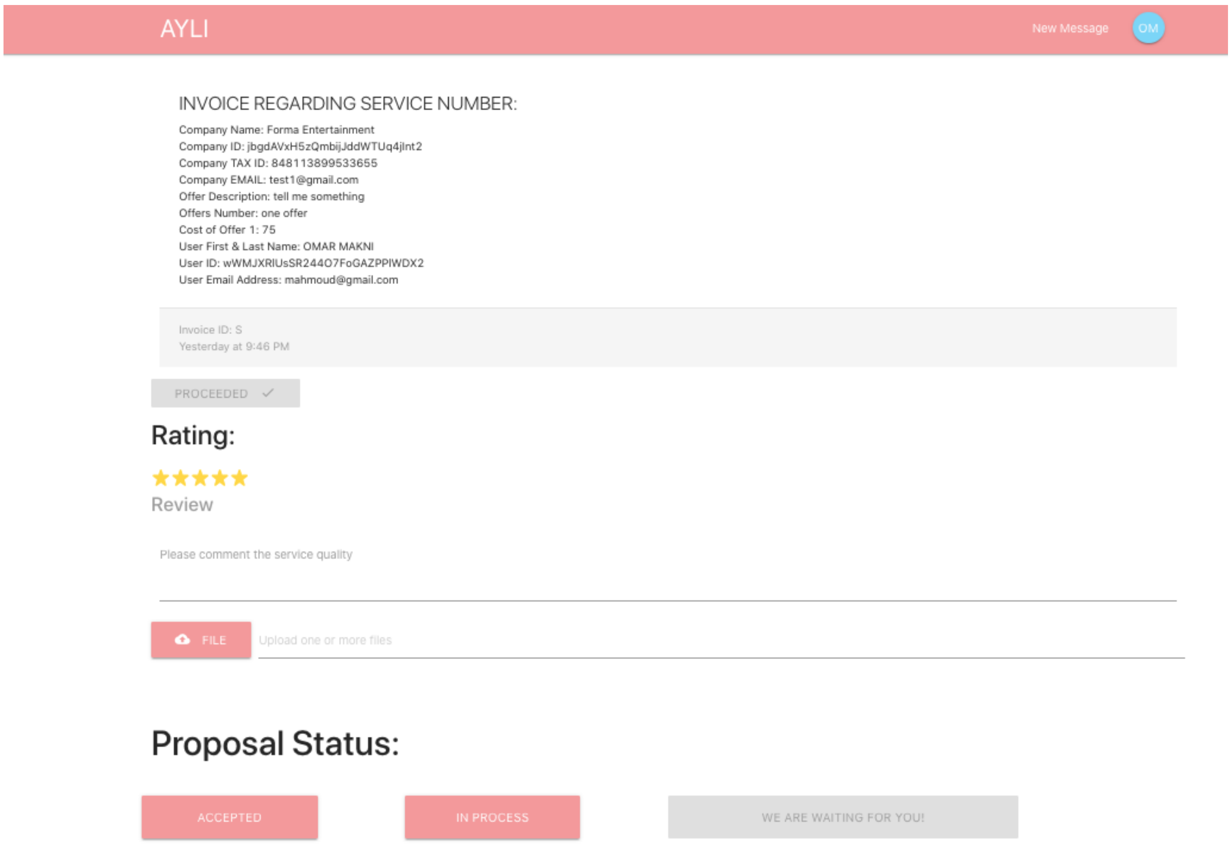
Figure 5 customer logged in with the map



Create new proposal

Offer Description

Figure 6 AYlI Creating Proposal



INVOICE REGARDING SERVICE NUMBER:
Company Name: Forma Entertainment
Company ID: jbgdAVxH5zQmbijJddWTUq4jlnt2
Company TAX ID: 848113899533655
Company EMAIL: test1@gmail.com
Offer Description: tell me something
Offers Number: one offer
Cost of Offer 1: 75
User First & Last Name: OMAR MAKNI
User ID: wWMJXRlUsSR244O7FoGAZPPIWDX2
User Email Address: mahmoud@gmail.com

Invoice ID: S
Yesterday at 9:46 PM

Figure 7 Customer paying the invoice and rating

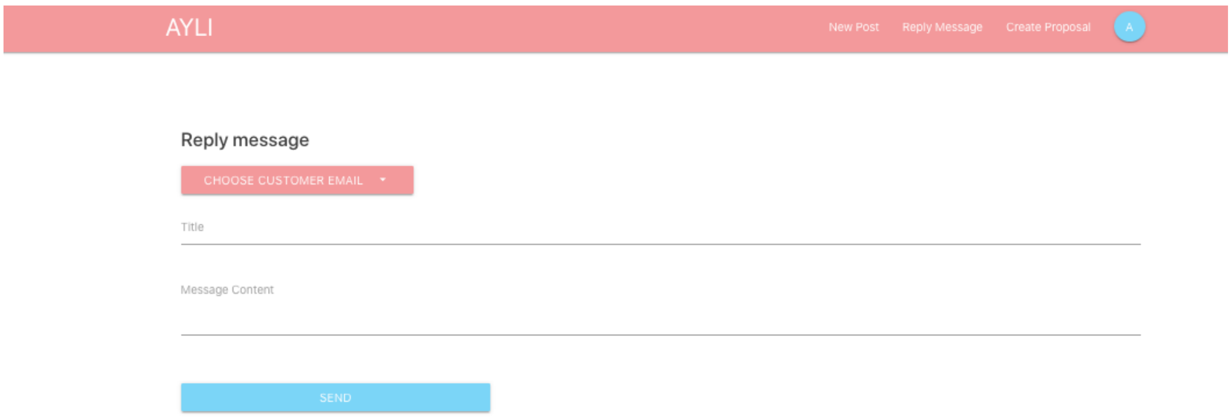
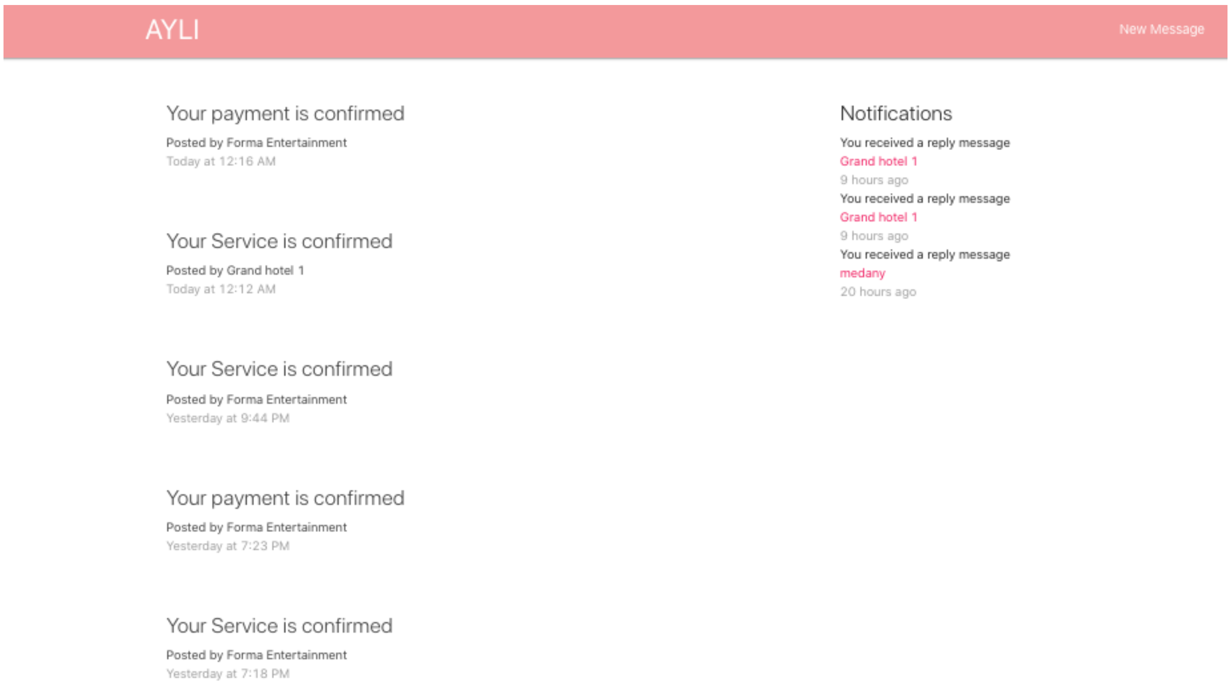


Figure 8 Replay for a message



Your payment is confirmed
Posted by Forma Entertainment
Today at 12:16 AM

Your Service is confirmed
Posted by Grand hotel 1
Today at 12:12 AM

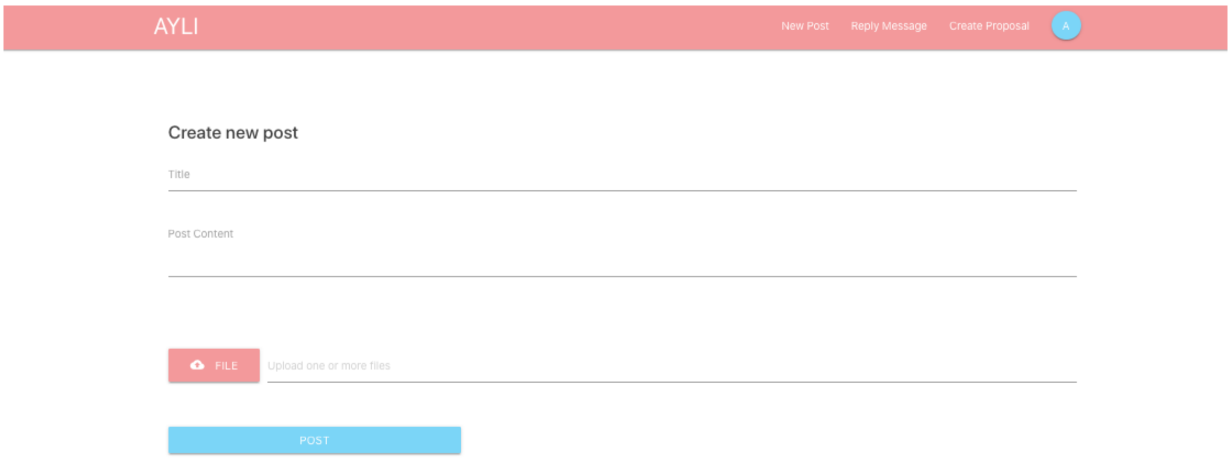
Your Service is confirmed
Posted by Forma Entertainment
Yesterday at 9:44 PM

Your payment is confirmed
Posted by Forma Entertainment
Yesterday at 7:23 PM

Your Service is confirmed
Posted by Forma Entertainment
Yesterday at 7:18 PM

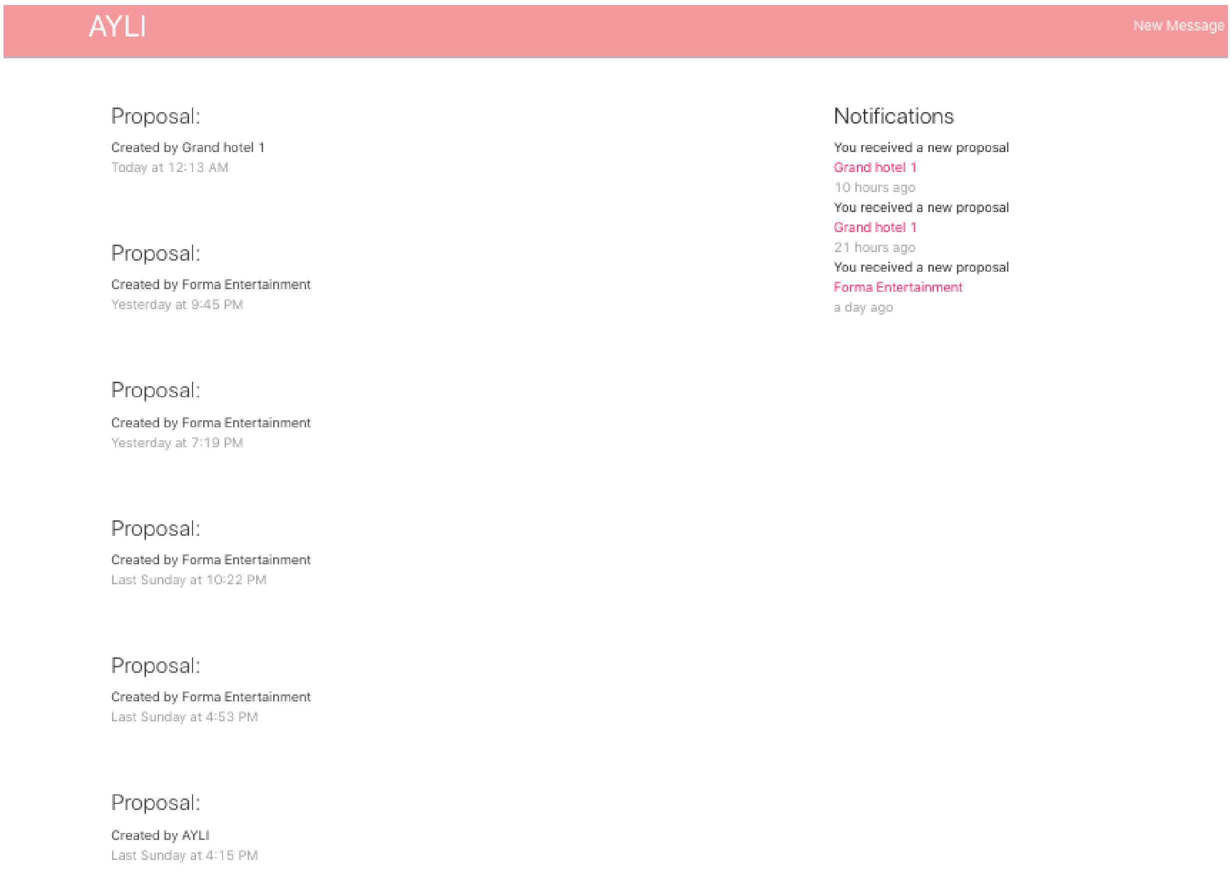
Notifications
You received a reply message
Grand hotel 1
9 hours ago
You received a reply message
Grand hotel 1
9 hours ago
You received a reply message
medany
20 hours ago

Figure 9 Inbox and notification



Create new post
Title
Post Content
FILE
Upload one or more files
POST

Figure 10 Create a broadcast by AYLI


AYLI

New Message

Proposal:

Created by Grand hotel 1

Today at 12:13 AM

Proposal:

Created by Forma Entertainment

Yesterday at 9:45 PM

Proposal:

Created by Forma Entertainment

Yesterday at 7:19 PM

Proposal:

Created by Forma Entertainment

Last Sunday at 10:22 PM

Proposal:

Created by Forma Entertainment

Last Sunday at 4:53 PM

Proposal:

Created by AYLI

Last Sunday at 4:15 PM

Notifications

You received a new proposal

Grand hotel 1

10 hours ago

You received a new proposal

Grand hotel 1

21 hours ago

You received a new proposal

Forma Entertainment

a day ago

Figure 11 Proposal List

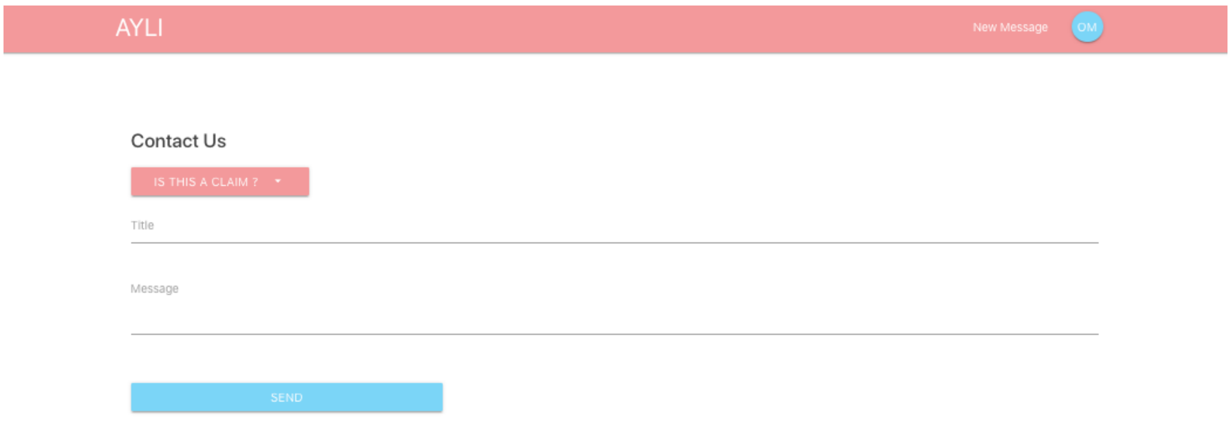


Figure 12 Submit a complain