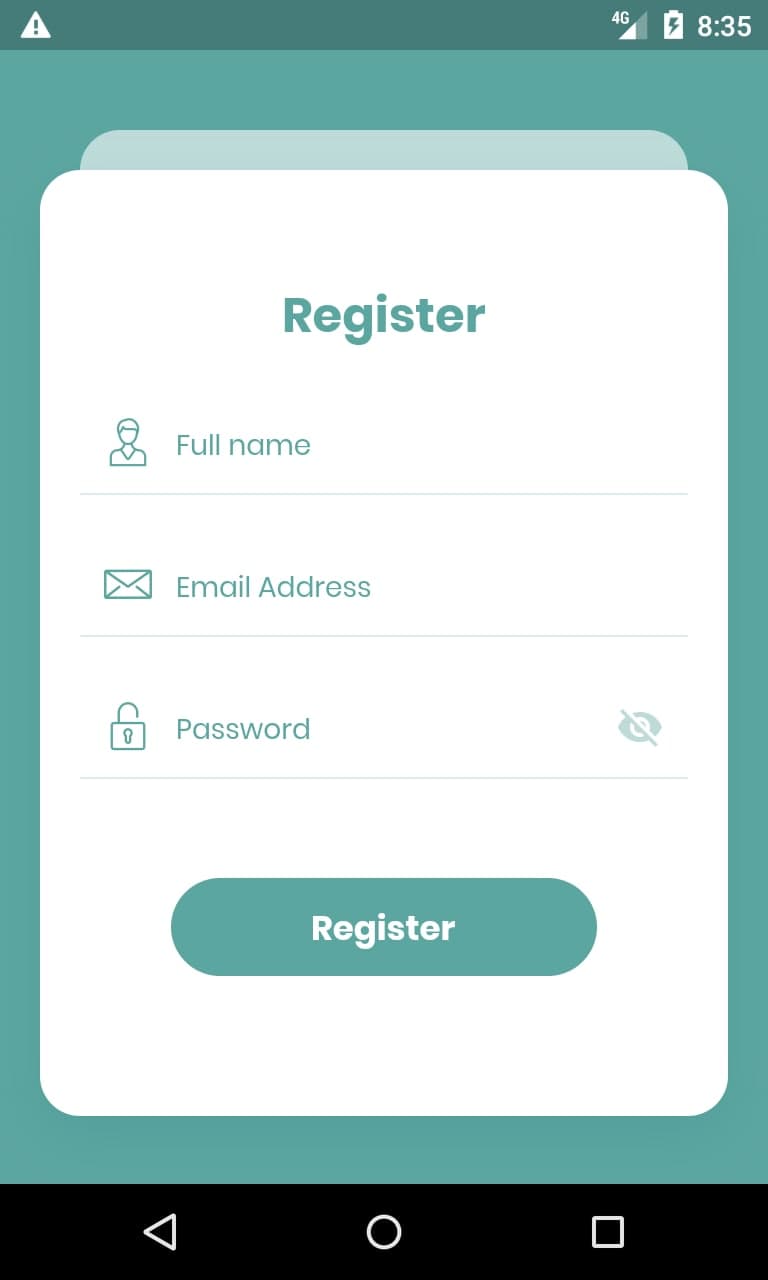
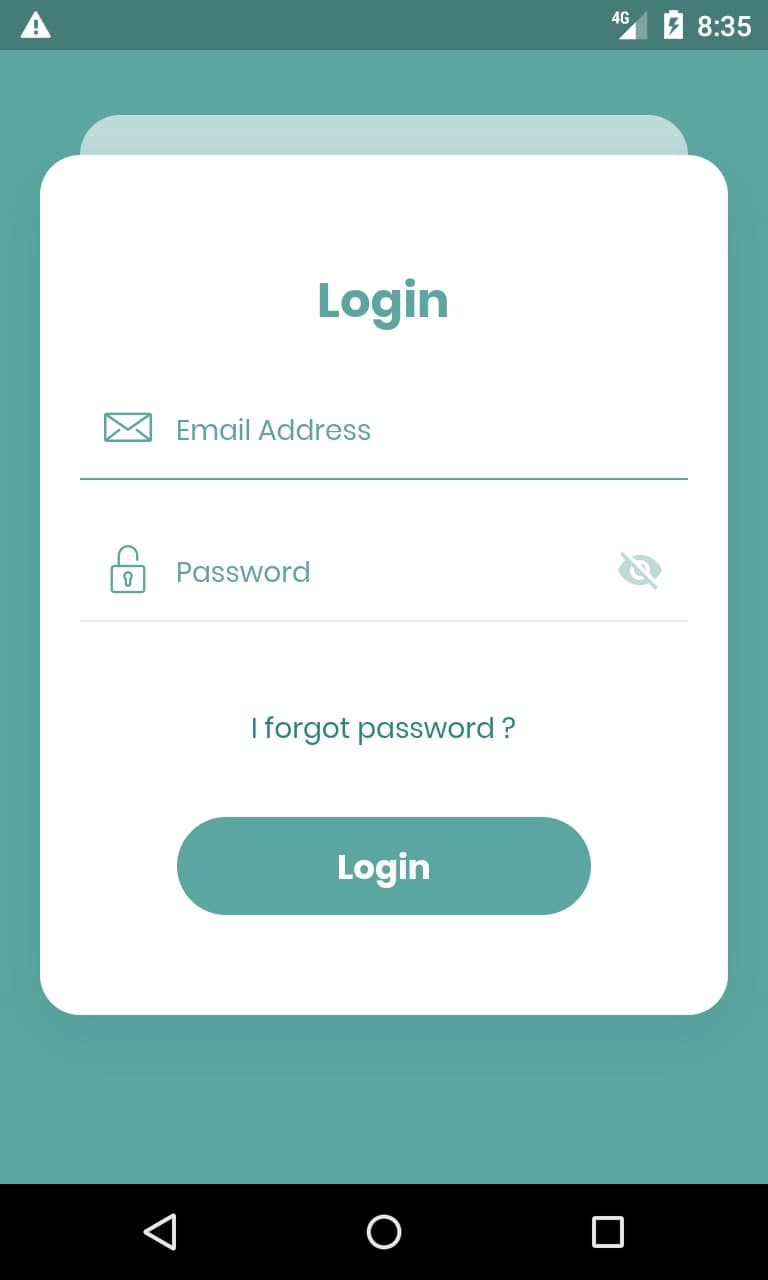
class SignUpWidget extends StatefulWidget {  
 @override  
 \_SignUpWidgetState createState() => \_SignUpWidgetState();  
}  
  
class \_SignUpWidgetState extends StateMVC<SignUpWidget> {  
 UserController \_con;  
  
 \_SignUpWidgetState() : super(UserController()) {  
 \_con = controller;  
 }  
  
 Widget build(BuildContext context) {  
 return Scaffold(  
 key: \_con.scaffoldKey,  
 backgroundColor: Theme.*of*(context).accentColor,  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Stack(  
 children: <Widget>[  
 Container(  
 width: double.*infinity*,  
 padding: EdgeInsets.symmetric(vertical: 30, horizontal: 20),  
 margin: EdgeInsets.symmetric(vertical: 65, horizontal: 40),  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(20),  
 color: Theme.*of*(context).primaryColor.withOpacity(0.6),  
 ),  
 ),  
 Container(  
 width: double.*infinity*,  
 padding: EdgeInsets.symmetric(vertical: 30, horizontal: 20),  
 margin: EdgeInsets.symmetric(vertical: 85, horizontal: 20),  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(20),  
 color: Theme.*of*(context).primaryColor,  
 boxShadow: [  
 BoxShadow(  
 color: Theme.*of*(context).hintColor.withOpacity(0.2), offset: Offset(0, 10), blurRadius: 20)  
 ],  
 ),  
 child: Form(  
 key: \_con.loginFormKey,  
 child: Column(  
 children: <Widget>[  
 SizedBox(height: 25),  
 Text(S.*of*(context).register, style: Theme.*of*(context).textTheme.display3),  
 SizedBox(height: 20),  
 new TextFormField(  
 style: TextStyle(color: Theme.*of*(context).accentColor),  
 keyboardType: TextInputType.*text*,  
 onSaved: (input) => \_con.user.name = input,  
 validator: (input) => input.length < 3 ? S.*of*(context).should\_be\_more\_than\_3\_letters : null,  
 decoration: InputDecoration(  
 contentPadding: EdgeInsets.all(12),  
 hintText: S.*of*(context).full\_name,  
 hintStyle: Theme.*of*(context).textTheme.body1.merge(  
 TextStyle(color: Theme.*of*(context).accentColor),  
 ),  
 prefixIcon: Icon(UiIcons.*user\_1*, color: Theme.*of*(context).accentColor),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Theme.*of*(context).accentColor.withOpacity(0.2))),  
 focusedBorder:  
 UnderlineInputBorder(borderSide: BorderSide(color: Theme.*of*(context).accentColor)),  
 ),  
 ),  
 SizedBox(height: 20),  
 new TextFormField(  
 style: TextStyle(color: Theme.*of*(context).accentColor),  
 keyboardType: TextInputType.*emailAddress*,  
 onSaved: (input) => \_con.user.email = input,  
 validator: (input) => !input.contains('@') ? S.*of*(context).should\_be\_a\_valid\_email : null,  
 decoration: new InputDecoration(  
 hintText: S.*of*(context).email\_address,  
 hintStyle: Theme.*of*(context).textTheme.body1.merge(  
 TextStyle(color: Theme.*of*(context).accentColor),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Theme.*of*(context).accentColor.withOpacity(0.2))),  
 focusedBorder:  
 UnderlineInputBorder(borderSide: BorderSide(color: Theme.*of*(context).accentColor)),  
 prefixIcon: Icon(  
 UiIcons.*envelope*,  
 color: Theme.*of*(context).accentColor,  
 ),  
 ),  
 ),  
 SizedBox(height: 20),  
 new TextFormField(  
 style: TextStyle(color: Theme.*of*(context).accentColor),  
 keyboardType: TextInputType.*text*,  
 obscureText: \_con.hidePassword,  
 onSaved: (input) => \_con.user.password = input,  
 validator: (input) => input.length < 3 ? S.*of*(context).should\_be\_more\_than\_3\_characters : null,  
 decoration: new InputDecoration(  
 hintText: S.*of*(context).password,  
 hintStyle: Theme.*of*(context).textTheme.body1.merge(  
 TextStyle(color: Theme.*of*(context).accentColor),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Theme.*of*(context).accentColor.withOpacity(0.2))),  
 focusedBorder:  
 UnderlineInputBorder(borderSide: BorderSide(color: Theme.*of*(context).accentColor)),  
 prefixIcon: Icon(  
 UiIcons.*padlock*,  
 color: Theme.*of*(context).accentColor,  
 ),  
 suffixIcon: IconButton(  
 onPressed: () {  
 setState(() {  
 \_con.hidePassword = !\_con.hidePassword;  
 });  
 },  
 color: Theme.*of*(context).accentColor.withOpacity(0.4),  
 icon: Icon(\_con.hidePassword ? Icons.*visibility\_off* : Icons.*visibility*),  
 ),  
 ),  
 ),  
 SizedBox(height: 50),  
   
 FlatButton(  
 padding: EdgeInsets.symmetric(vertical: 12, horizontal: 70),  
 onPressed: () {  
 \_con.register();  
 },  
 child: Text(  
 S.*of*(context).register,  
 style: Theme.*of*(context).textTheme.title.merge(  
 TextStyle(color: Theme.*of*(context).primaryColor),  
 ),  
 ),  
 color: Theme.*of*(context).accentColor,  
 shape: StadiumBorder(),  
 ),  
 SizedBox(height: 40),  
 ],  
 ),  
 ),  
 ),  
 ],  
 ),  
 FlatButton(  
 onPressed: () {  
 Navigator.*of*(context).pushNamed('/Login');  
 },  
 child: RichText(  
 text: TextSpan(  
 style: Theme.*of*(context).textTheme.title.merge(  
 TextStyle(color: Theme.*of*(context).primaryColor,fontSize: 16,fontWeight: FontWeight.*w600*),  
 ),  
 children: [  
 TextSpan(text: S.*of*(context).i\_have\_account\_back\_to\_login),  
 ],  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

signup screen



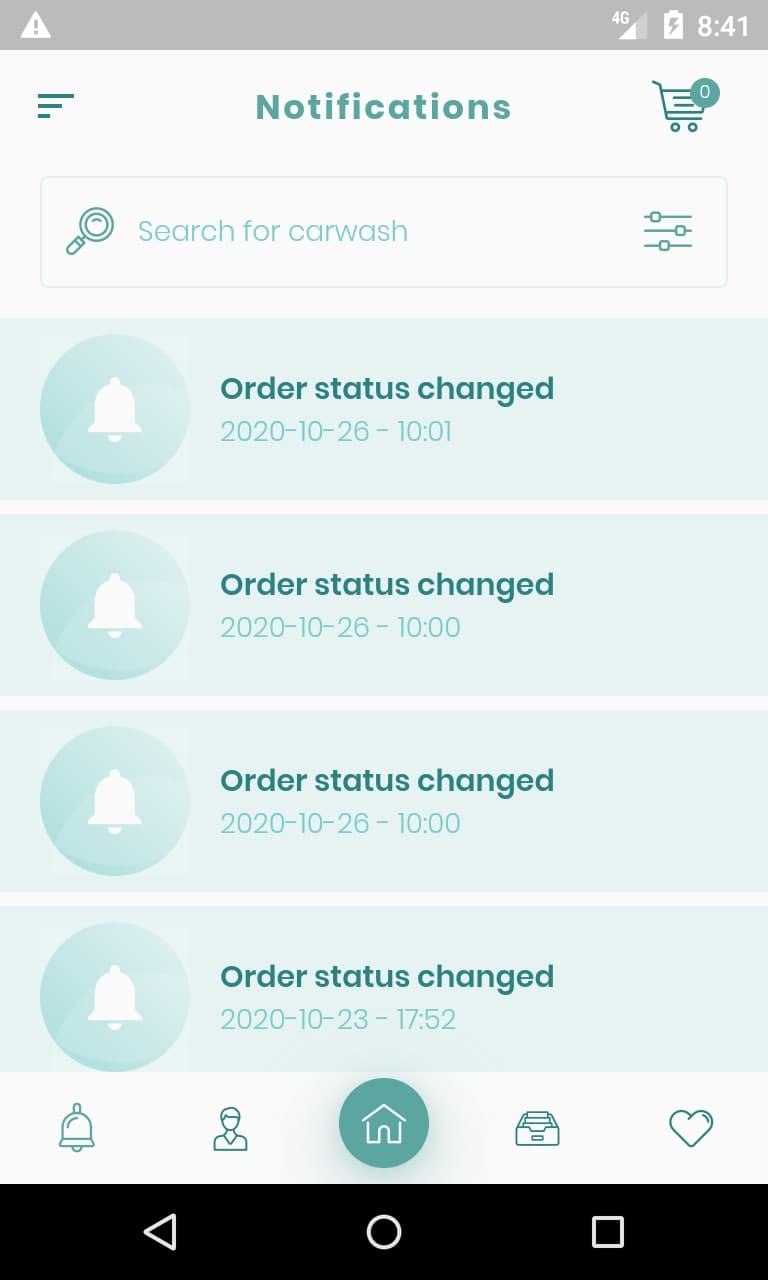
class SignInWidget extends StatefulWidget {  
 @override  
 \_SignInWidgetState createState() => \_SignInWidgetState();  
}  
  
class \_SignInWidgetState extends StateMVC<SignInWidget> {  
 UserController \_con;  
  
 \_SignInWidgetState() : super(UserController()) {  
 \_con = controller;  
 }  
 @override  
 void initState() {  
 super.initState();  
 if (userRepo.currentUser.value.apiToken != null) {  
 Navigator.*of*(context).pushReplacementNamed('/Pages', arguments: 2);  
 }  
 }  
  
 Widget build(BuildContext context) {  
 return Scaffold(  
 key: \_con.scaffoldKey,  
 backgroundColor: Theme.*of*(context).accentColor,  
 body: SingleChildScrollView(  
 child: Column(  
 children: <Widget>[  
 Stack(  
 children: <Widget>[  
 Container(  
 width: double.*infinity*,  
 padding: EdgeInsets.symmetric(vertical: 30, horizontal: 20),  
 margin: EdgeInsets.symmetric(vertical: 65, horizontal: 40),  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(20),  
 color: Theme.*of*(context).primaryColor.withOpacity(0.6),  
 ),  
 ),  
 Container(  
 width: double.*infinity*,  
 padding: EdgeInsets.symmetric(vertical: 30, horizontal: 20),  
 margin: EdgeInsets.symmetric(vertical: 85, horizontal: 20),  
 decoration: BoxDecoration(  
 borderRadius: BorderRadius.circular(20),  
 color: Theme.*of*(context).primaryColor,  
 boxShadow: [  
 BoxShadow(  
 color: Theme.*of*(context).hintColor.withOpacity(0.2), offset: Offset(0, 10), blurRadius: 20)  
 ]),  
 child: Form(  
 key: \_con.loginFormKey,  
 child: Column(  
 children: <Widget>[  
 SizedBox(height: 25),  
 Text(  
 S.*of*(context).login,  
 style: Theme.*of*(context).textTheme.display3),  
 SizedBox(height: 20),  
 new TextFormField(  
 style: TextStyle(color: Theme.*of*(context).accentColor),  
 keyboardType: TextInputType.*emailAddress*,  
 onSaved: (input) => \_con.user.email = input,  
 validator: (input) => !input.contains('@') ? S.*of*(context).should\_be\_a\_valid\_email : null,  
 decoration: new InputDecoration(  
 hintText: S.*of*(context).email\_address,  
 hintStyle: Theme.*of*(context).textTheme.body1.merge(  
 TextStyle(color: Theme.*of*(context).accentColor),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Theme.*of*(context).accentColor.withOpacity(0.2))),  
 focusedBorder:  
 UnderlineInputBorder(borderSide: BorderSide(color: Theme.*of*(context).accentColor)),  
 prefixIcon: Icon(  
 UiIcons.*envelope*,  
 color: Theme.*of*(context).accentColor,  
 ),  
 ),  
 ),  
 SizedBox(height: 20),  
 new TextFormField(  
 style: TextStyle(color: Theme.*of*(context).accentColor),  
 keyboardType: TextInputType.*text*,  
 onSaved: (input) => \_con.user.password = input,  
 validator: (input) => input.length < 3 ? S.*of*(context).should\_be\_more\_than\_3\_characters : null,  
 obscureText: \_con.hidePassword,  
 decoration: new InputDecoration(  
 hintText: S.*of*(context).password,  
 hintStyle: Theme.*of*(context).textTheme.body1.merge(  
 TextStyle(color: Theme.*of*(context).accentColor),  
 ),  
 enabledBorder: UnderlineInputBorder(  
 borderSide: BorderSide(color: Theme.*of*(context).accentColor.withOpacity(0.2))),  
 focusedBorder:  
 UnderlineInputBorder(borderSide: BorderSide(color: Theme.*of*(context).accentColor)),  
 prefixIcon: Icon(  
 UiIcons.*padlock*,  
 color: Theme.*of*(context).accentColor,  
 ),  
 suffixIcon: IconButton(  
 onPressed: () {  
 setState(() {  
 \_con.hidePassword = !\_con.hidePassword;  
 });  
 },  
 color: Theme.*of*(context).accentColor.withOpacity(0.4),  
 icon: Icon(\_con.hidePassword ? Icons.*visibility\_off* : Icons.*visibility*),  
 ),  
 ),  
 ),  
 SizedBox(height: 30),  
 FlatButton(  
 onPressed: () {  
 Navigator.*of*(context).pushReplacementNamed('/ForgetPassword');  
 },  
 child: Text(  
 S.*of*(context).i\_forgot\_password,  
 style: Theme.*of*(context).textTheme.body1,  
 ),  
 ),  
 SizedBox(height: 20),  
 FlatButton(  
 padding: EdgeInsets.symmetric(vertical: 12, horizontal: 80),  
 onPressed: () {  
 \_con.login();  
 },  
 child: Text(  
 S.*of*(context).login,  
 style: Theme.*of*(context).textTheme.title.merge(  
 TextStyle(color: Theme.*of*(context).primaryColor),  
 ),  
 ),  
 color: Theme.*of*(context).accentColor,  
 shape: StadiumBorder(),  
 ),  
 SizedBox(height: 15),  
 SizedBox(height: 5),  
 ],  
 ),  
 ),  
 ),  
 ],  
 ),  
 FlatButton(  
 onPressed: () {  
 Navigator.*of*(context).pushNamed('/SignUp');  
 },  
 child: RichText(  
 text: TextSpan(  
 style: Theme.*of*(context).textTheme.title.merge(  
 TextStyle(color: Theme.*of*(context).primaryColor,fontSize: 16,fontWeight: FontWeight.*w600*),  
 ),  
 children: [  
 TextSpan(text: S.*of*(context).i\_dont\_have\_an\_account),  
 ],  
 ),  
 ),  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

signin screen



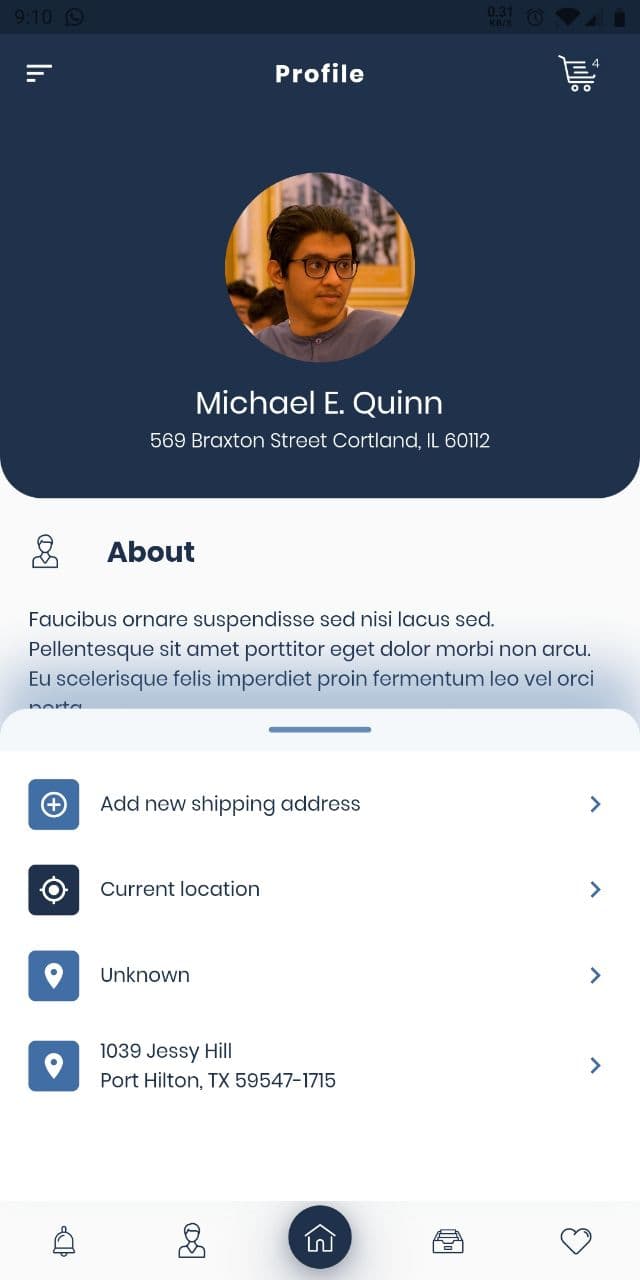
class NotificationsWidget extends StatefulWidget {  
  
 final GlobalKey<ScaffoldState> parentScaffoldKey;  
  
 const NotificationsWidget({Key key, this.parentScaffoldKey}) : super(key: key);  
 @override  
 \_NotificationsWidgetState createState() => \_NotificationsWidgetState();  
}  
  
class \_NotificationsWidgetState extends StateMVC<NotificationsWidget> {  
 NotificationController \_con;  
  
 \_NotificationsWidgetState() : super(NotificationController()){  
 \_con =controller;  
 }  
  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 key:\_con.scaffoldKey,  
 appBar: AppBar(  
 automaticallyImplyLeading: false,  
 leading: new IconButton(  
 icon: new Icon(Icons.*sort*, color: Theme.*of*(context).hintColor),  
 onPressed: () => widget.parentScaffoldKey.currentState.openDrawer(),  
 ),  
 backgroundColor: Colors.*transparent*,  
 elevation: 0,  
 centerTitle: true,  
 title: ValueListenableBuilder(  
 valueListenable: settingsRepo.setting,  
 builder: (context, value, child) {  
 return Text(  
 S.*of*(context).notifications,  
 style: Theme.*of*(context).textTheme.title.merge(TextStyle(letterSpacing: 1.3)),  
 );  
 },  
 ),  
 actions: <Widget>[  
 new ShoppingCartButtonWidget(  
 iconColor: Theme.*of*(context).hintColor, labelColor: Theme.*of*(context).accentColor  
 ),  
 ],  
 ),  
 body: currentUser.value.apiToken == null  
 ? PermissionDeniedWidget()  
 : RefreshIndicator(  
 onRefresh: \_con.refreshNotifications,  
 child:SingleChildScrollView(  
 padding: EdgeInsets.symmetric(vertical: 7),  
 child: Column(  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 20),  
 child: SearchBarWidget(  
 onClickFilter: (event) {  
 widget.parentScaffoldKey.currentState.openEndDrawer();  
 },  
 ),  
 ),  
 Offstage(  
 offstage: \_con.notifications.isEmpty,  
 child: ListView.separated(  
 padding: EdgeInsets.symmetric(vertical: 15),  
 shrinkWrap: true,  
 primary: false,  
 itemCount: \_con.notifications.length,  
 separatorBuilder: (context, index) {  
 return SizedBox(height: 7);  
 },  
 itemBuilder: (context, index) {  
 return NotificationItemWidget(  
 notification: \_con.notifications.elementAt(index),  
 onDismissed: (notification) {  
 setState(() {  
 \_con.removeFromNotification(\_con.notifications.elementAt(index));  
 });  
 },  
 );  
 },  
 ),  
 ),  
 Offstage(  
 offstage: \_con.notifications.isNotEmpty,  
 child: EmptyNotificationsWidget(),  
 )  
 ],  
 ),  
 ),  
 )  
 );  
 }  
}

notification screen



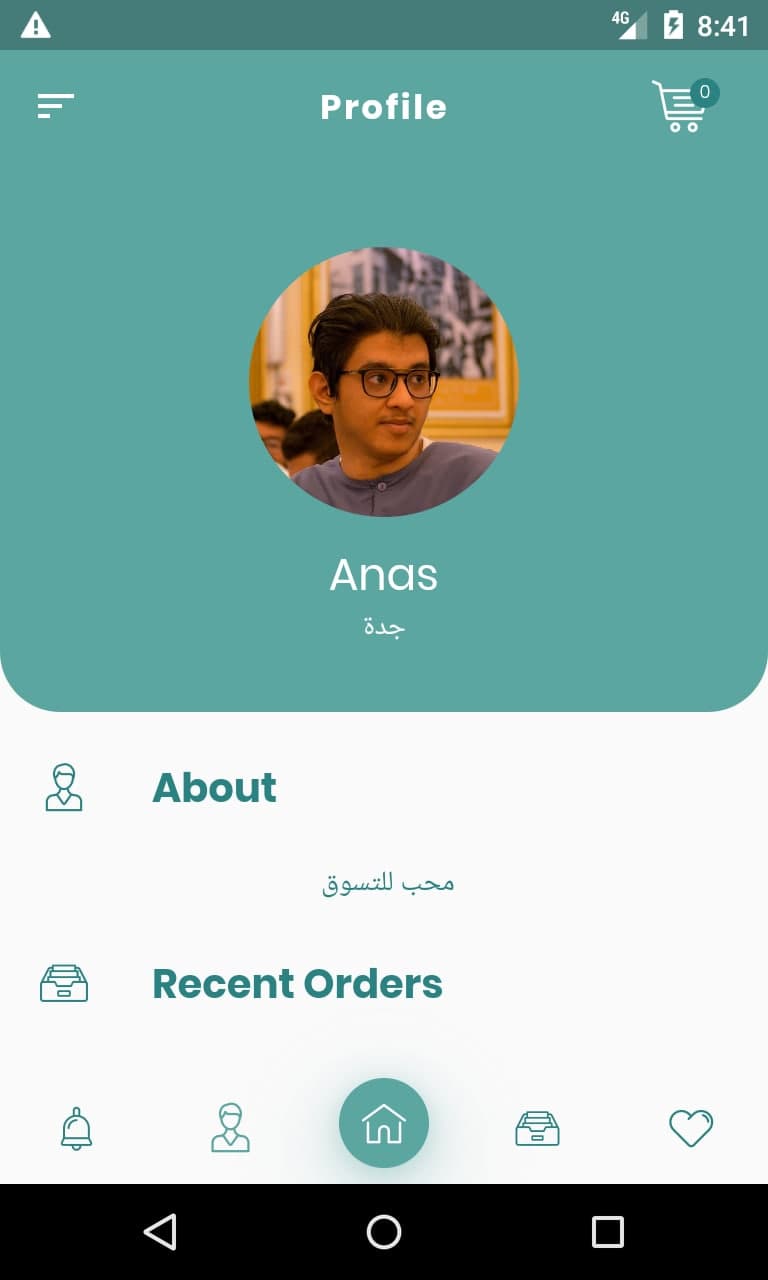
class DeliveryAddressesWidget extends StatefulWidget {  
 final RouteArgument routeArgument;  
  
 DeliveryAddressesWidget({Key key, this.routeArgument}) : super(key: key);  
  
 @override  
 \_DeliveryAddressesWidgetState createState() => \_DeliveryAddressesWidgetState();  
}  
  
class \_DeliveryAddressesWidgetState extends StateMVC<DeliveryAddressesWidget> {  
 DeliveryAddressesController \_con;  
 PaymentMethodList list;  
  
 \_DeliveryAddressesWidgetState() : super(DeliveryAddressesController()) {  
 \_con = controller;  
 }  
  
 @override  
 void initState() {  
 list = new PaymentMethodList();  
 super.initState();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 key: \_con.scaffoldKey,  
 appBar: AppBar(  
 backgroundColor: Colors.*transparent*,  
 elevation: 0,  
 centerTitle: true,  
 leading: new IconButton(  
 icon: new Icon(UiIcons.*return\_icon*,  
 color: Theme.*of*(context).hintColor),  
 onPressed: () => Navigator.*of*(context).pop(),  
 ),  
 title: Text(  
 S.*of*(context).delivery\_addresses,  
 style: Theme.*of*(context).textTheme.title.merge(TextStyle(letterSpacing: 1.3)),  
 ),  
 actions: <Widget>[  
 new ShoppingCartButtonWidget(iconColor: Theme.*of*(context).hintColor, labelColor: Theme.*of*(context).accentColor),  
 ],  
 ),  
 floatingActionButton:// \_con.cart != null && \_con.cart.product.store.availableForDelivery  
 FloatingActionButton(  
 onPressed: () async {  
 LocationResult result = await showLocationPicker(  
 context,  
 setting.value.googleMapsKey,  
 initialCenter: LatLng(deliveryAddress.value?.latitude ?? 0, deliveryAddress.value?.longitude ?? 0),  
 //automaticallyAnimateToCurrentLocation: true,  
 //mapStylePath: 'assets/mapStyle.json',  
 myLocationButtonEnabled: true,  
 //resultCardAlignment: Alignment.bottomCenter,  
 );  
 \_con.addAddress(new Address.fromJSON({  
 'address': result.address,  
 'latitude': result.latLng.latitude,  
 'longitude': result.latLng.longitude,  
 }));  
 print("result = $result");  
 //setState(() => \_pickedLocation = result);  
 },  
 backgroundColor: Theme.*of*(context).accentColor,  
 child: Icon(  
 Icons.*add*,  
 color: Theme.*of*(context).primaryColor,  
 )),  
 //: SizedBox(height: 0),  
 body: RefreshIndicator(  
 onRefresh: \_con.refreshAddresses,  
 child: SingleChildScrollView(  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 mainAxisAlignment: MainAxisAlignment.start,  
 mainAxisSize: MainAxisSize.max,  
 children: <Widget>[  
 Padding(  
 padding: const EdgeInsets.only(top: 10, left: 20, right: 20),  
 child: ListTile(  
 contentPadding: EdgeInsets.symmetric(vertical: 0),  
 leading: Icon(  
 Icons.*map*,  
 color: Theme.*of*(context).hintColor,  
 ),  
 title: Text(  
 S.*of*(context).delivery\_addresses,  
 maxLines: 1,  
 overflow: TextOverflow.ellipsis,  
 style: Theme.*of*(context).textTheme.display1,  
 ),  
 subtitle: Text(  
 S.*of*(context).long\_press\_to\_edit\_item\_swipe\_item\_to\_delete\_it,  
 maxLines: 2,  
 overflow: TextOverflow.ellipsis,  
 style: Theme.*of*(context).textTheme.caption,  
 ),  
 ),  
 ),  
  
 \_con.addresses.isEmpty  
 ? CircularLoadingWidget(height: 250)  
 : ListView.separated(  
 padding: EdgeInsets.symmetric(vertical: 15),  
 scrollDirection: Axis.vertical,  
 shrinkWrap: true,  
 primary: false,  
 itemCount: \_con.addresses.length,  
 separatorBuilder: (context, index) {  
 return SizedBox(height: 15);  
 },  
 itemBuilder: (context, index) {  
 return DeliveryAddressesItemWidget(  
 address: \_con.addresses.elementAt(index),  
 onPressed: (Address \_address) {  
 DeliveryAddressDialog(  
 context: context,  
 address: \_address,  
 onChanged: (Address \_address) {  
 \_con.updateAddress(\_address);  
 },  
 );  
 },  
 onLongPress: (Address \_address) {  
 DeliveryAddressDialog(  
 context: context,  
 address: \_address,  
 onChanged: (Address \_address) {  
 \_con.updateAddress(\_address);  
 },  
 );  
 },  
 onDismissed: (Address \_address) {  
 \_con.removeDeliveryAddress(\_address);  
 },  
 );  
 },  
 ),  
 ],  
 ),  
 ),  
 ),  
 );  
 }  
}

addresses screen



class AccountWidget extends StatefulWidget {  
 final GlobalKey<ScaffoldState> parentScaffoldKey;  
  
 AccountWidget({Key key, this.parentScaffoldKey}) : super(key: key);  
 @override  
 \_AccountWidgetState createState() => \_AccountWidgetState();  
}  
  
class \_AccountWidgetState extends StateMVC<AccountWidget> {  
 ProfileController \_con;  
  
 \_AccountWidgetState() : super(ProfileController()) {  
 \_con = controller;  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 final theme = Theme.*of*(context).copyWith(dividerColor: Colors.*transparent*);  
 return Scaffold(  
 key: \_con.scaffoldKey,  
 drawer: DrawerWidget(),  
 appBar: AppBar(  
 automaticallyImplyLeading: false,  
 leading: new IconButton(  
 icon: new Icon(Icons.*sort*, color: currentUser.value.apiToken == null?Theme.*of*(context).hintColor:Theme.*of*(context).primaryColor),  
 onPressed: () => widget.parentScaffoldKey.currentState.openDrawer(),  
 ),  
  
 backgroundColor: currentUser.value.apiToken == null?Colors.*transparent*:Theme.*of*(context).accentColor,  
 elevation: 0,  
 centerTitle: true,  
 title: Text(  
 S.*of*(context).profile,  
 style: currentUser.value.apiToken == null  
 ?Theme.*of*(context)  
 .textTheme  
 .title  
 .merge(TextStyle(letterSpacing: 1.3))  
 :Theme.*of*(context)  
 .textTheme  
 .title  
 .merge(TextStyle(letterSpacing: 1.3, color:Theme.*of*(context).primaryColor)),  
 ),  
 actions: <Widget>[  
 new ShoppingCartButtonWidget(  
 iconColor: currentUser.value.apiToken == null?Theme.*of*(context).hintColor:Theme.*of*(context).primaryColor,  
 labelColor: currentUser.value.apiToken == null?Theme.*of*(context).accentColor:Theme.*of*(context).hintColor),  
 ],  
 ),  
 body: currentUser.value.apiToken == null  
 ? PermissionDeniedWidget()  
 : SingleChildScrollView(  
// padding: const EdgeInsets.symmetric(horizontal: 0, vertical: 10),  
 child: Column(  
 children: <Widget>[  
 ProfileAvatarWidget(user: currentUser.value),  
 ListTile(  
 contentPadding: EdgeInsets.symmetric(horizontal: 20, vertical: 10),  
 leading: Icon(  
 UiIcons.*user\_1*,  
 color: Theme.*of*(context).hintColor,  
 ),  
 title: Text(  
 S.*of*(context).about,  
 style: Theme.*of*(context).textTheme.display1,  
 ),  
 ),  
 Padding(  
 padding: const EdgeInsets.symmetric(horizontal: 20),  
 child: Text(  
 currentUser.value?.bio ?? "",  
 style: Theme.*of*(context).textTheme.body1,  
 ),  
 ),  
 ListTile(  
 contentPadding: EdgeInsets.symmetric(horizontal: 20, vertical: 10),  
 leading: Icon(  
 UiIcons.*inbox*,  
 color: Theme.*of*(context).hintColor,  
 ),  
 title: Text(  
 S.*of*(context).recent\_orders,  
 style: Theme.*of*(context).textTheme.display1,  
 ),  
 ),  
 \_con.recentOrders.isEmpty  
 ? CircularLoadingWidget(height: 200)  
 : ListView.builder(  
 scrollDirection: Axis.vertical,  
 shrinkWrap: true,  
 primary: false,  
 itemCount: \_con.recentOrders.length,  
 itemBuilder: (context, index) {  
 return Theme(  
 data: theme,  
 child: ExpansionTile(  
 initiallyExpanded: true,  
 title: Row(  
 children: <Widget>[  
 Expanded(  
 child: Text(  
 '${S.*of*(context).order\_id}: #${\_con.recentOrders.elementAt(index).id}')),  
 Text(  
 '${\_con.recentOrders.elementAt(index).orderStatus.status}',  
 style: Theme.*of*(context).textTheme.caption,  
 ),  
 ],  
 ),  
 children: List.generate(\_con.recentOrders.elementAt(index).productOrders.length,  
 (indexProduct) {  
 return OrderItemWidget(  
 heroTag: 'recent\_orders',  
 order: \_con.recentOrders.elementAt(index),  
 productOrder:  
 \_con.recentOrders.elementAt(index).productOrders.elementAt(indexProduct));  
 }),  
 ),  
 );  
 },  
 ),  
 ],  
 ),  
 ),  
 );  
 }  
}

account profile



class LoginController extends Controller  
{  
 /\*  
 |--------------------------------------------------------------------------  
 | Login Controller  
 |--------------------------------------------------------------------------  
 |  
 | This controller handles authenticating users for the application and  
 | redirecting them to your home screen. The controller uses a trait  
 | to conveniently provide its functionality to your applications.  
 |  
 \*/  
  
 use AuthenticatesUsers;  
  
 */\*\**  
 *\* Where to redirect users after login.*  
 *\**  
 *\** ***@var*** *string*  
 *\*/*  
protected $redirectTo = '/';  
 private $userRepository;  
 private $uploadRepository;  
 private $roleRepository;  
  
 */\*\**  
 *\* Create a new controller instance.*  
 *\**  
 *\** ***@return*** *void*  
 *\*/*  
public function \_\_construct(UserRepository $userRepository, UploadRepository $uploadRepository, RoleRepository $roleRepository)  
 {  
 $this->middleware('guest')->except('logout');  
 $this->userRepository = $userRepository;  
 $this->uploadRepository = $uploadRepository;  
 $this->roleRepository = $roleRepository;  
 }  
  
  
}

login controller

class RegisterController extends Controller  
{  
 /\*  
 |--------------------------------------------------------------------------  
 | Register Controller  
 |--------------------------------------------------------------------------  
 |  
 | This controller handles the registration of new users as well as their  
 | validation and creation. By default this controller uses a trait to  
 | provide this functionality without requiring any additional code.  
 |  
 \*/  
  
 use RegistersUsers;  
  
 */\*\**  
 *\* Where to redirect users after registration.*  
 *\**  
 *\** ***@var*** *string*  
 *\*/*  
protected $redirectTo = '/';  
  
 private $userRepository;  
 private $uploadRepository;  
 private $roleRepository;  
  
 */\*\**  
 *\* Create a new controller instance.*  
 *\**  
 *\** ***@return*** *void*  
 *\*/*  
public function \_\_construct(UserRepository $userRepository, UploadRepository $uploadRepository, RoleRepository $roleRepository)  
 {  
 $this->middleware('guest');  
 $this->userRepository = $userRepository;  
 $this->uploadRepository = $uploadRepository;  
 $this->roleRepository = $roleRepository;  
 }  
  
 */\*\**  
 *\* Get a validator for an incoming registration request.*  
 *\**  
 *\** ***@param*** *array $data*  
 *\** ***@return*** *\Illuminate\Contracts\Validation\Validator*  
 *\*/*  
protected function validator(array $data)  
 {  
 return Validator::make($data, [  
 'name' => 'required|string|max:255',  
 'email' => 'required|string|email|max:255|unique:users',  
 'password' => 'required|string|min:6|confirmed',  
 ]);  
 }  
  
 */\*\**  
 *\* Create a new user instance after a valid registration.*  
 *\**  
 *\** ***@param*** *array $data*  
 *\** ***@return***  
*\*/*  
protected function create(array $data)  
 {  
 $user = new User;  
 $user->name = $data['name'];  
 $user->email = $data['email'];  
 $user->password = Hash::make($data['password']);  
 $user->api\_token = str\_random(60);  
 $user->save();  
  
 $defaultRoles = $this->roleRepository->findByField('default', '1');  
 $defaultRoles = $defaultRoles->pluck('name')->toArray();  
 $user->assignRole($defaultRoles);  
  
 return $user;  
 }  
}

register controller

class NotificationController extends Controller  
{  
 */\*\** ***@var*** *NotificationRepository \*/*  
private $notificationRepository;  
  
 */\*\**  
 *\** ***@var*** *CustomFieldRepository*  
 *\*/*  
private $customFieldRepository;  
  
 private $userRepository;  
  
 public function \_\_construct(NotificationRepository $notificationRepo, CustomFieldRepository $customFieldRepo,  
 UserRepository $userRepo)  
 {  
 parent::*\_\_construct*();  
 $this->notificationRepository = $notificationRepo;  
 $this->customFieldRepository = $customFieldRepo;  
 $this->userRepository = $userRepo;  
 }  
  
 */\*\**  
 *\* Display a listing of the Notification.*  
 *\**  
 *\** ***@param*** *NotificationDataTable $notificationDataTable*  
 *\** ***@return*** *Response*  
 *\*/*  
public function index(NotificationDataTable $notificationDataTable)  
 {  
 return $notificationDataTable->render('notifications.index');  
 }  
  
 */\*\**  
 *\* Show the form for creating a new Notification.*  
 *\**  
 *\** ***@return*** *Response*  
 *\*/*  
public function create()  
 {  
 $user = $this->userRepository->pluck('name', 'id');  
  
 $hasCustomField = in\_array($this->notificationRepository->model(), setting('custom\_field\_models', []));  
 if ($hasCustomField) {  
 $customFields = $this->customFieldRepository->findByField('custom\_field\_model', $this->notificationRepository->model());  
 $html = generateCustomField($customFields);  
 }  
 return view('notifications.create')->with("customFields", isset($html) ? $html : false)->with("user", $user);  
 }  
  
 */\*\**  
 *\* Store a newly created Notification in storage.*  
 *\**  
 *\** ***@param*** *CreateNotificationRequest $request*  
 *\**  
 *\** ***@return*** *Response*  
 *\*/*  
public function store(CreateNotificationRequest $request)  
 {  
 $input = $request->all();  
 $customFields = $this->customFieldRepository->findByField('custom\_field\_model', $this->notificationRepository->model());  
 try {  
 $notification = $this->notificationRepository->create($input);  
 $notification->customFieldsValues()->createMany(getCustomFieldsValues($customFields, $request));  
  
 } catch (ValidatorException $e) {  
 Flash::*error*($e->getMessage());  
 }  
  
 Flash::*success*(\_\_('lang.saved\_successfully', ['operator' => \_\_('lang.notification')]));  
  
 return redirect(route('notifications.index'));  
 }  
  
 */\*\**  
 *\* Display the specified Notification.*  
 *\**  
 *\** ***@param*** *int $id*  
 *\**  
 *\** ***@return*** *Response*  
 *\*/*  
public function show($id)  
 {  
 $notification = $this->notificationRepository->findWithoutFail($id);  
  
 if (empty($notification)) {  
 Flash::*error*('Notification not found');  
  
 return redirect(route('notifications.index'));  
 }  
 try {  
 //dd((new Carbon('now'))->format('Y-m-d H:i:s'));  
 $notification = $this->notificationRepository->update(['read\_at' => (new Carbon())], $id);  
  
 } catch (ValidatorException $e) {  
 Flash::*error*($e->getMessage());  
 } catch (\Exception $e) {  
 Flash::*error*($e->getMessage());  
 }  
  
 return redirect(route('notifications.index'));  
 }  
  
 */\*\**  
 *\* Show the form for editing the specified Notification.*  
 *\**  
 *\** ***@param*** *int $id*  
 *\**  
 *\** ***@return*** *Response*  
 *\*/*  
public function edit($id)  
 {  
 $notification = $this->notificationRepository->findWithoutFail($id);  
 $user = $this->userRepository->pluck('name', 'id');  
  
  
 if (empty($notification)) {  
 Flash::*error*(\_\_('lang.not\_found', ['operator' => \_\_('lang.notification')]));  
  
 return redirect(route('notifications.index'));  
 }  
 $customFieldsValues = $notification->customFieldsValues()->with('customField')->get();  
 $customFields = $this->customFieldRepository->findByField('custom\_field\_model', $this->notificationRepository->model());  
 $hasCustomField = in\_array($this->notificationRepository->model(), setting('custom\_field\_models', []));  
 if ($hasCustomField) {  
 $html = generateCustomField($customFields, $customFieldsValues);  
 }  
  
 return view('notifications.edit')->with('notification', $notification)->with("customFields", isset($html) ? $html : false)->with("user", $user);  
 }  
  
 */\*\**  
 *\* Update the specified Notification in storage.*  
 *\**  
 *\** ***@param*** *int $id*  
 *\** ***@param*** *UpdateNotificationRequest $request*  
 *\**  
 *\** ***@return*** *Response*  
 *\*/*  
public function update($id, UpdateNotificationRequest $request)  
 {  
 $notification = $this->notificationRepository->findWithoutFail($id);  
  
 if (empty($notification)) {  
 Flash::*error*('Notification not found');  
 return redirect(route('notifications.index'));  
 }  
 $input = $request->all();  
 $customFields = $this->customFieldRepository->findByField('custom\_field\_model', $this->notificationRepository->model());  
 try {  
 $notification = $this->notificationRepository->update($input, $id);  
 foreach (getCustomFieldsValues($customFields, $request) as $value) {  
 $notification->customFieldsValues()  
 ->updateOrCreate(['custom\_field\_id' => $value['custom\_field\_id']], $value);  
 }  
 } catch (ValidatorException $e) {  
 Flash::*error*($e->getMessage());  
 }  
  
 Flash::*success*(\_\_('lang.updated\_successfully', ['operator' => \_\_('lang.notification')]));  
  
 return redirect(route('notifications.index'));  
 }  
  
 */\*\**  
 *\* Remove the specified Notification from storage.*  
 *\**  
 *\** ***@param*** *int $id*  
 *\**  
 *\** ***@return*** *Response*  
 *\*/*  
public function destroy($id)  
 {  
 $notification = $this->notificationRepository->findWithoutFail($id);  
  
 if (empty($notification)) {  
 Flash::*error*('Notification not found');  
  
 return redirect(route('notifications.index'));  
 }  
  
 $this->notificationRepository->delete($id);  
  
 Flash::*success*(\_\_('lang.deleted\_successfully', ['operator' => \_\_('lang.notification')]));  
  
 return redirect(route('notifications.index'));  
 }  
  
 */\*\**  
 *\* Remove Media of Notification*  
 *\** ***@param*** *Request $request*  
 *\*/*  
public function removeMedia(Request $request)  
 {  
 $input = $request->all();  
 $notification = $this->notificationRepository->findWithoutFail($input['id']);  
 try {  
 if ($notification->hasMedia($input['collection'])) {  
 $notification->getFirstMedia($input['collection'])->delete();  
 }  
 } catch (\Exception $e) {  
 Log::error($e->getMessage());  
 }  
 }  
}

notification controller

class DeliveryAddressController extends Controller{ */\*\** ***@var*** *DeliveryAddressRepository \*/*private $deliveryAddressRepository; */\*\** *\** ***@var*** *CustomFieldRepository* *\*/*private $customFieldRepository; */\*\** *\** ***@var*** *UserRepository* *\*/*private $userRepository; public function \_\_construct(DeliveryAddressRepository $deliveryAddressRepo, CustomFieldRepository $customFieldRepo , UserRepository $userRepo) { parent::*\_\_construct*(); $this->deliveryAddressRepository = $deliveryAddressRepo; $this->customFieldRepository = $customFieldRepo; $this->userRepository = $userRepo; } */\*\** *\* Display a listing of the DeliveryAddress.* *\** *\** ***@param*** *DeliveryAddressDataTable $deliveryAddressDataTable* *\** ***@return*** *Response* *\*/*public function index(DeliveryAddressDataTable $deliveryAddressDataTable) { return $deliveryAddressDataTable->render('delivery\_addresses.index'); } */\*\** *\* Show the form for creating a new DeliveryAddress.* *\** *\** ***@return*** *Response* *\*/*public function create() { $user = $this->userRepository->pluck('name','id');  $hasCustomField = in\_array($this->deliveryAddressRepository->model(),setting('custom\_field\_models',[])); if($hasCustomField){ $customFields = $this->customFieldRepository->findByField('custom\_field\_model', $this->deliveryAddressRepository->model()); $html = generateCustomField($customFields); } return view('delivery\_addresses.create')->with("customFields", isset($html) ? $html : false)->with("user",$user); } */\*\** *\* Store a newly created DeliveryAddress in storage.* *\** *\** ***@param*** *CreateDeliveryAddressRequest $request* *\** *\** ***@return*** *Response* *\*/*public function store(CreateDeliveryAddressRequest $request) { $input = $request->all(); $customFields = $this->customFieldRepository->findByField('custom\_field\_model', $this->deliveryAddressRepository->model()); try { $deliveryAddress = $this->deliveryAddressRepository->create($input); $deliveryAddress->customFieldsValues()->createMany(getCustomFieldsValues($customFields,$request));  } catch (ValidatorException $e) { Flash::*error*($e->getMessage()); } Flash::*success*(\_\_('lang.saved\_successfully',['operator' => \_\_('lang.delivery\_address')])); return redirect(route('deliveryAddresses.index')); } */\*\** *\* Display the specified DeliveryAddress.* *\** *\** ***@param*** *int $id* *\** *\** ***@return*** *Response* *\*/*public function show($id) { $deliveryAddress = $this->deliveryAddressRepository->findWithoutFail($id); if (empty($deliveryAddress)) { Flash::*error*('Delivery Address not found'); return redirect(route('deliveryAddresses.index')); } return view('delivery\_addresses.show')->with('deliveryAddress', $deliveryAddress); } */\*\** *\* Show the form for editing the specified DeliveryAddress.* *\** *\** ***@param*** *int $id* *\** *\** ***@return*** *Response* *\*/*public function edit($id) { $deliveryAddress = $this->deliveryAddressRepository->findWithoutFail($id); $user = $this->userRepository->pluck('name','id');  if (empty($deliveryAddress)) { Flash::*error*(\_\_('lang.not\_found',['operator' => \_\_('lang.delivery\_address')])); return redirect(route('deliveryAddresses.index')); } $customFieldsValues = $deliveryAddress->customFieldsValues()->with('customField')->get(); $customFields = $this->customFieldRepository->findByField('custom\_field\_model', $this->deliveryAddressRepository->model()); $hasCustomField = in\_array($this->deliveryAddressRepository->model(),setting('custom\_field\_models',[])); if($hasCustomField) { $html = generateCustomField($customFields, $customFieldsValues); } return view('delivery\_addresses.edit')->with('deliveryAddress', $deliveryAddress)->with("customFields", isset($html) ? $html : false)->with("user",$user); } */\*\** *\* Update the specified DeliveryAddress in storage.* *\** *\** ***@param*** *int $id* *\** ***@param*** *UpdateDeliveryAddressRequest $request* *\** *\** ***@return*** *Response* *\*/*public function update($id, UpdateDeliveryAddressRequest $request) { $deliveryAddress = $this->deliveryAddressRepository->findWithoutFail($id); if (empty($deliveryAddress)) { Flash::*error*('Delivery Address not found'); return redirect(route('deliveryAddresses.index')); } $input = $request->all(); $customFields = $this->customFieldRepository->findByField('custom\_field\_model', $this->deliveryAddressRepository->model()); try { $deliveryAddress = $this->deliveryAddressRepository->update($input, $id);   foreach (getCustomFieldsValues($customFields, $request) as $value){ $deliveryAddress->customFieldsValues() ->updateOrCreate(['custom\_field\_id'=>$value['custom\_field\_id']],$value); } } catch (ValidatorException $e) { Flash::*error*($e->getMessage()); } Flash::*success*(\_\_('lang.updated\_successfully',['operator' => \_\_('lang.delivery\_address')])); return redirect(route('deliveryAddresses.index')); } */\*\** *\* Remove the specified DeliveryAddress from storage.* *\** *\** ***@param*** *int $id* *\** *\** ***@return*** *Response* *\*/*public function destroy($id) { $deliveryAddress = $this->deliveryAddressRepository->findWithoutFail($id); if (empty($deliveryAddress)) { Flash::*error*('Delivery Address not found'); return redirect(route('deliveryAddresses.index')); } $this->deliveryAddressRepository->delete($id); Flash::*success*(\_\_('lang.deleted\_successfully',['operator' => \_\_('lang.delivery\_address')])); return redirect(route('deliveryAddresses.index')); } */\*\** *\* Remove Media of DeliveryAddress* *\** ***@param*** *Request $request* *\*/*public function removeMedia(Request $request) { $input = $request->all(); $deliveryAddress = $this->deliveryAddressRepository->findWithoutFail($input['id']); try { if($deliveryAddress->hasMedia($input['collection'])){ $deliveryAddress->getFirstMedia($input['collection'])->delete(); } } catch (\Exception $e) { Log::error($e->getMessage()); } }}

**delivery address controller**

*/\*\**  
 *\* Class User*  
 *\** ***@package*** *App\Models*  
 *\** ***@version*** *July 10, 2018, 11:44 am UTC*  
 *\**  
 *\** ***@property*** *string name*  
 *\** ***@property*** *string email*  
 *\** ***@property*** *string password*  
 *\** ***@property*** *string api\_token*  
 *\** ***@property*** *string device\_token*  
 *\*/*  
class User extends *Authenticatable* implements HasMedia  
{  
 use Notifiable;  
 use Billable;  
 use HasMediaTrait {  
 getFirstMediaUrl as protected getFirstMediaUrlTrait;  
 }  
 use HasRoles;  
  
 */\*\**  
 *\* Validation rules*  
 *\**  
 *\** ***@var*** *array*  
 *\*/*  
public static *$rules* = [  
 'name' => 'required|string|max:255',  
 'email' => 'required|string|email|max:255|unique:users',  
 // 'password' => 'required',  
 ];  
 public $table = 'users';  
 */\*\**  
 *\* The attributes that are mass assignable.*  
 *\**  
 *\** ***@var*** *array*  
 *\*/*  
public $fillable = [  
 'name',  
 'email',  
 'password',  
 'api\_token',  
 'device\_token',  
 ];  
 */\*\**  
 *\* The attributes that should be casted to native types.*  
 *\**  
 *\** ***@var*** *array*  
 *\*/*  
protected $casts = [  
 'name' => 'string',  
 'email' => 'string',  
 'password' => 'string',  
 'api\_token' => 'string',  
 'device\_token' => 'string',  
 'remember\_token' => 'string'  
 ];  
 */\*\**  
 *\* New Attributes*  
 *\**  
 *\** ***@var*** *array*  
 *\*/*  
protected $appends = [  
 'custom\_fields',  
 'has\_media'  
 ];  
  
 */\*\**  
 *\* The attributes that should be hidden for arrays.*  
 *\**  
 *\** ***@var*** *array*  
 *\*/*  
protected $hidden = [  
 'password', 'remember\_token',  
 ];  
  
 */\*\**  
 *\* Specifies the user's FCM token*  
 *\**  
 *\** ***@return*** *string*  
 *\*/*  
public function routeNotificationForFcm($notification)  
 {  
 return $this->device\_token;  
 }  
  
 */\*\**  
 *\** ***@param*** *Media|null $media*  
 *\** ***@throws*** *\Spatie\Image\Exceptions\InvalidManipulation*  
 *\*/*  
public function registerMediaConversions(\Spatie\MediaLibrary\Models\Media $media = null)  
 {  
 $this->addMediaConversion('thumb')  
 ->fit(Manipulations::*FIT\_CROP*, 200, 200)  
 ->sharpen(10);  
  
 $this->addMediaConversion('icon')  
 ->fit(Manipulations::*FIT\_CROP*, 100, 100)  
 ->sharpen(10);  
 }  
  
 */\*\**  
 *\* to generate media url in case of fallback will*  
 *\* return the file type icon*  
 *\** ***@param*** *string $conversion*  
 *\** ***@return*** *string url*  
 *\*/*  
public function getFirstMediaUrl($collectionName = 'default', $conversion = '')  
 {  
 $url = $this->getFirstMediaUrlTrait($collectionName);  
 if ($url) {  
 $array = explode('.', $url);  
 $extension = strtolower(end($array));  
 if (in\_array($extension, config('medialibrary.extensions\_has\_thumb'))) {  
 return asset($this->getFirstMediaUrlTrait($collectionName, $conversion));  
 } else {  
 return asset(config('medialibrary.icons\_folder') . '/' . $extension . '.png');  
 }  
 }else{  
 return asset('images/avatar\_default.png');  
 }  
 }  
  
 public function getCustomFieldsAttribute()  
 {  
 $hasCustomField = in\_array(static::class, setting('custom\_field\_models', []));  
 if (!$hasCustomField) {  
 return [];  
 }  
 $array = $this->customFieldsValues()  
 ->join('custom\_fields', 'custom\_fields.id', '=', 'custom\_field\_values.custom\_field\_id')  
// ->where('custom\_fields.in\_table', '=', true)  
 ->select(['value', 'view', 'name'])  
 ->get()->toArray();  
  
 return convertToAssoc($array, 'name');  
 }  
  
 public function customFieldsValues()  
 {  
 return $this->morphMany('App\Models\CustomFieldValue', 'customizable');  
 }  
  
 */\*\**  
 *\* Add Media to api results*  
 *\** ***@return*** *bool*  
 *\*/*  
public function getHasMediaAttribute()  
 {  
 return $this->hasMedia('avatar') ? true : false;  
 }  
  
 */\*\**  
 *\** ***@return*** *\Illuminate\Database\Eloquent\Relations\BelongsToMany*  
 *\*\*/*  
public function stores()  
 {  
 return $this->belongsToMany(\App\Models\Store::class, 'user\_stores');  
 }

user model

*\*/*class DeliveryAddress extends *Model*{ public $table = 'delivery\_addresses';  public $fillable = [ 'description', 'address', 'latitude', 'longitude', 'is\_default', 'user\_id' ]; */\*\** *\* The attributes that should be casted to native types.* *\** *\** ***@var*** *array* *\*/*protected $casts = [ 'description' => 'string', 'address' => 'string', 'latitude' => 'double', 'longitude' => 'double', 'is\_default' => 'boolean', 'user\_id' => 'integer' ]; */\*\** *\* Validation rules* *\** *\** ***@var*** *array* *\*/*public static *$rules* = [ 'description' => 'required', 'address' => 'required', 'user\_id' => 'required|exists:users,id' ]; */\*\** *\* New Attributes* *\** *\** ***@var*** *array* *\*/*protected $appends = [ 'custom\_fields',  ]; public function customFieldsValues() { return $this->morphMany('App\Models\CustomFieldValue', 'customizable'); } public function getCustomFieldsAttribute() { $hasCustomField = in\_array(static::class,setting('custom\_field\_models',[])); if (!$hasCustomField){ return []; } $array = $this->customFieldsValues() ->join('custom\_fields','custom\_fields.id','=','custom\_field\_values.custom\_field\_id') ->where('custom\_fields.in\_table','=',true) ->get()->toArray(); return convertToAssoc($array,'name'); } */\*\** *\** ***@return*** *\Illuminate\Database\Eloquent\Relations\BelongsTo* *\*\*/*public function user() { return $this->belongsTo(\App\Models\User::class, 'user\_id', 'id'); }

address model

class Notification extends *Model*{ public $table = 'notifications'; protected $primaryKey = 'id'; // or null public $incrementing = false;  public $fillable = [ 'type', 'read\_at' ]; */\*\** *\* The attributes that should be casted to native types.* *\** *\** ***@var*** *array* *\*/*protected $casts = [ 'type' => 'string', 'read\_at' => 'date' ]; */\*\** *\* Validation rules* *\** *\** ***@var*** *array* *\*/*public static *$rules* = [ 'type' => 'required', ]; */\*\** *\* New Attributes* *\** *\** ***@var*** *array* *\*/*protected $appends = [ 'custom\_fields',  ]; public function customFieldsValues() { return $this->morphMany('App\Models\CustomFieldValue', 'customizable'); } public function getCustomFieldsAttribute() { $hasCustomField = in\_array(static::class,setting('custom\_field\_models',[])); if (!$hasCustomField){ return []; } $array = $this->customFieldsValues() ->join('custom\_fields','custom\_fields.id','=','custom\_field\_values.custom\_field\_id') ->where('custom\_fields.in\_table','=',true) ->get()->toArray(); return convertToAssoc($array,'name'); } */\*\** *\** ***@return*** *\Illuminate\Database\Eloquent\Relations\BelongsTo* *\*\*/*public function user() { return $this->belongsTo(\App\Models\User::class, 'notifiable\_id', 'id'); } }

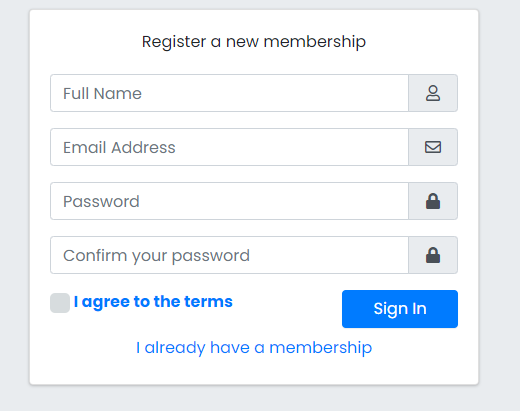
notification model

<!DOCTYPE html><html lang="{{setting('language','app()->getLocale()')}}"><head> <meta charset="utf-8"> <meta http-equiv="X-UA-Compatible" content="IE=edge"> <title>{{setting('app\_name')}} | {{setting('app\_short\_description')}}</title> <!-- Tell the browser to be responsive to screen width --> <meta name="viewport" content="width=device-width, initial-scale=1"> <link rel="icon" type="image/png" href="{{$app\_logo}}" /> <!-- Font Awesome --> <link rel="stylesheet" href="{{asset('plugins/font-awesome/css/font-awesome.min.css')}}"> <!-- Ionicons --> <link href="https://unpkg.com/ionicons@4.1.2/dist/css/ionicons.min.css" rel="stylesheet"> <script src="https://kit.fontawesome.com/f129a6342d.js" crossorigin="anonymous"></script> <!-- Theme style --> <link rel="stylesheet" href="{{asset('dist/css/adminlte.css')}}"> @if(app()->getLocale()=='ar') <link rel="stylesheet" href="{{asset('dist/css/rtl/adminlte.css')}}"> @endif <!-- iCheck --> <link rel="stylesheet" href="{{asset('plugins/iCheck/flat/blue.css')}}">{{--<!-- Bootstrap -->--}} <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">{{--<link rel="stylesheet" href="{{asset('plugins/bootstrap/css/bootstrap.min.css')}}">--}}<!-- Google Font: Poppins --> <link href="https://fonts.googleapis.com/css?family=Poppins:300,400,400i,700" rel="stylesheet"> <link rel="stylesheet" href="{{asset('css/custom.css')}}"> @if(app()->getLocale()=='ar') <link rel="stylesheet" href="{{asset('dist/css/rtl/custom.css')}}"> @endif</head><body class="hold-transition login-page"><div class="login-box"> <div class="login-logo"> <a href="{{ url('/') }}"><img src="{{$app\_logo}}" alt="{{setting('app\_name')}}"></a> </div> <div class="card"> @yield('register\_bar') </div> <!-- /.login-logo --> <div class="card"> @yield('content') </div></div><!-- /.login-box --><!-- script --><script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script><script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script><script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV" crossorigin="anonymous"></script><!-- jQuery --><script src="{{asset('plugins/jquery/jquery.min.js')}}"></script><!-- Bootstrap 4 --><script src="{{asset('plugins/bootstrap/js/bootstrap.bundle.min.js')}}"></script><!-- iCheck --><script src="{{asset('plugins/iCheck/icheck.min.js')}}"></script><script> $(function () { $('.icheck input').iCheck({ checkboxClass: 'icheckbox\_flat-blue', radioClass: 'iradio\_flat-blue', increaseArea: '20%' // optional }) })</script>@yield('script')</body></html>

**default blade page**

@extends('layouts.auth.default')@section('content') <div class="card-body login-card-body"> <p class="login-box-msg">{{\_\_('auth.register\_new\_member')}}</p> <form action="{{ url('/register') }}" method="post"> {!! csrf\_field() !!} <div class="input-group mb-3"> <input value="{{ old('name') }}" type="name" class="form-control {{ $errors->has('name') ? ' is-invalid' : '' }}" name="name" placeholder="{{\_\_('auth.name')}}" aria-label="{{\_\_('auth.name')}}"> <div class="input-group-append"> <span class="input-group-text"><i class="fa fa-user"></i></span> </div> @if ($errors->has('name')) <div class="invalid-feedback"> {{ $errors->first('name') }} </div> @endif </div> <div class="input-group mb-3"> <input value="{{ old('email') }}" type="email" class="form-control {{ $errors->has('email') ? ' is-invalid' : '' }}" name="email" placeholder="{{\_\_('auth.email')}}" aria-label="{{\_\_('auth.email')}}"> <div class="input-group-append"> <span class="input-group-text"><i class="fa fa-envelope"></i></span> </div> @if ($errors->has('email')) <div class="invalid-feedback"> {{ $errors->first('email') }} </div> @endif </div> <div class="input-group mb-3"> <input value="{{ old('password') }}" type="password" class="form-control {{ $errors->has('password') ? ' is-invalid' : '' }}" name="password" placeholder="{{\_\_('auth.password')}}" aria-label="{{\_\_('auth.password')}}"> <div class="input-group-append"> <span class="input-group-text"><i class="fa fa-lock"></i></span> </div> @if ($errors->has('password')) <div class="invalid-feedback"> {{ $errors->first('password') }} </div> @endif </div> <div class="input-group mb-3"> <input value="{{ old('password\_confirmation') }}" type="password" class="form-control {{ $errors->has('password\_confirmation') ? ' is-invalid' : '' }}" name="password\_confirmation" placeholder="{{\_\_('auth.password\_confirmation')}}" aria-label="{{\_\_('auth.password\_confirmation')}}"> <div class="input-group-append"> <span class="input-group-text"><i class="fa fa-lock"></i></span> </div> @if ($errors->has('password\_confirmation')) <div class="invalid-feedback"> {{ $errors->first('password\_confirmation') }} </div> @endif </div> <div class="row mb-2"> <div class="col-8"> <div class="checkbox icheck"> <label > <input type="checkbox" name="remember"> <a href="{{ url('/terms') }}" >{{\_\_('auth.agree')}} </a></label> </div> </div> <!-- /.col --> <div class="col-4"> <button type="submit" class="btn btn-primary btn-block">{{\_\_('auth.register')}}</button> </div> <!-- /.col --> </div> </form> @if(setting('enable\_facebook',false) || setting('enable\_google',false) || setting('enable\_twitter',false)) <div class="social-auth-links text-center mb-3"> <p style="text-transform: uppercase">- {{\_\_('lang.or')}} -</p> @if(setting('enable\_facebook',false)) <a href="{{url('login/facebook')}}" class="btn btn-block btn-facebook"> <i class="fa fa-facebook mr-2"></i> {{\_\_('auth.login\_facebook')}} </a> @endif @if(setting('enable\_google',false)) <a href="{{url('login/google')}}" class="btn btn-block btn-google"> <i class="fa fa-google-plus mr-2"></i> {{\_\_('auth.login\_google')}} </a> @endif @if(setting('enable\_twitter',false)) <a href="{{url('login/twitter')}}" class="btn btn-block btn-twitter"> <i class="fa fa-twitter mr-2"></i> {{\_\_('auth.login\_twitter')}} </a> @endif </div> <!-- /.social-auth-links --> @endif <p class="mb-1 text-center"> <a href="{{ url('/login') }}">{{\_\_('auth.already\_member')}}</a> </p> </div> <!-- /.login-card-body -->@endsection

**register page**

****

@extends('layouts.auth.default')  
@section('content')  
 <div class="card-body login-card-body">  
 <div class="card-body login-card-body">  
 <p class="login-box-msg">{{\_\_('auth.login\_title')}}</p>  
  
 <form action="{{ url('/login') }}" method="post">  
 {!! csrf\_field() !!}  
  
 <div class="input-group mb-3">  
 <input value="{{ old('email') }}" type="email" class="form-control {{ $errors->has('email') ? ' is-invalid' : '' }}" name="email" placeholder="{{ \_\_('auth.email') }}" aria-label="{{ \_\_('auth.email') }}">  
 <div class="input-group-append">  
 <span class="input-group-text"><i class="fa fa-envelope"></i></span>  
 </div>  
 @if ($errors->has('email'))  
 <div class="invalid-feedback">  
 {{ $errors->first('email') }}  
 </div>  
 @endif  
 </div>  
  
 <div class="input-group mb-3">  
 <input value="{{ old('password') }}" type="password" class="form-control {{ $errors->has('password') ? ' is-invalid' : '' }}" name="password" placeholder="{{\_\_('auth.password')}}" aria-label="{{\_\_('auth.password')}}">  
 <div class="input-group-append">  
 <span class="input-group-text"><i class="fa fa-lock"></i></span>  
 </div>  
 @if ($errors->has('password'))  
 <div class="invalid-feedback">  
 {{ $errors->first('password') }}  
 </div>  
 @endif  
 </div>  
  
 <div class="row mb-2">  
 <div class="col-8">  
 <div class="checkbox icheck">  
 <label> <input type="checkbox" name="remember"> {{\_\_('auth.remember\_me')}}  
 </label>  
 </div>  
 </div>  
 <!-- /.col -->  
 <div class="col-4">  
 <button type="submit" class="btn btn-primary btn-block">{{\_\_('auth.login')}}</button>  
 </div>  
 <!-- /.col -->  
 </div>  
 </form>  
  
 <p class="mb-1 text-center">  
 <a href="{{ url('/password/reset') }}">{{\_\_('auth.forgot\_password')}}</a>  
 </p>  
 <p class="mb-0 text-center">  
 <a href="{{ url('/register') }}" class="text-center">{{\_\_('auth.register\_new\_member')}}</a>  
 </p>  
 <div class="col-4" style="margin-left: auto; margin-right: auto; padding-top: 20px">  
 <button id="switchLanguage" type="submit" class="btn btn-secondary btn-block">{{\_\_('auth.login.lang')}}</button>  
 </div>  
 </div>  
 </div>  
@endsection

login page

