

Group B

Software Engineering & Databases

H.E.M.W.I.G.D.G. Ekanayake (1074)

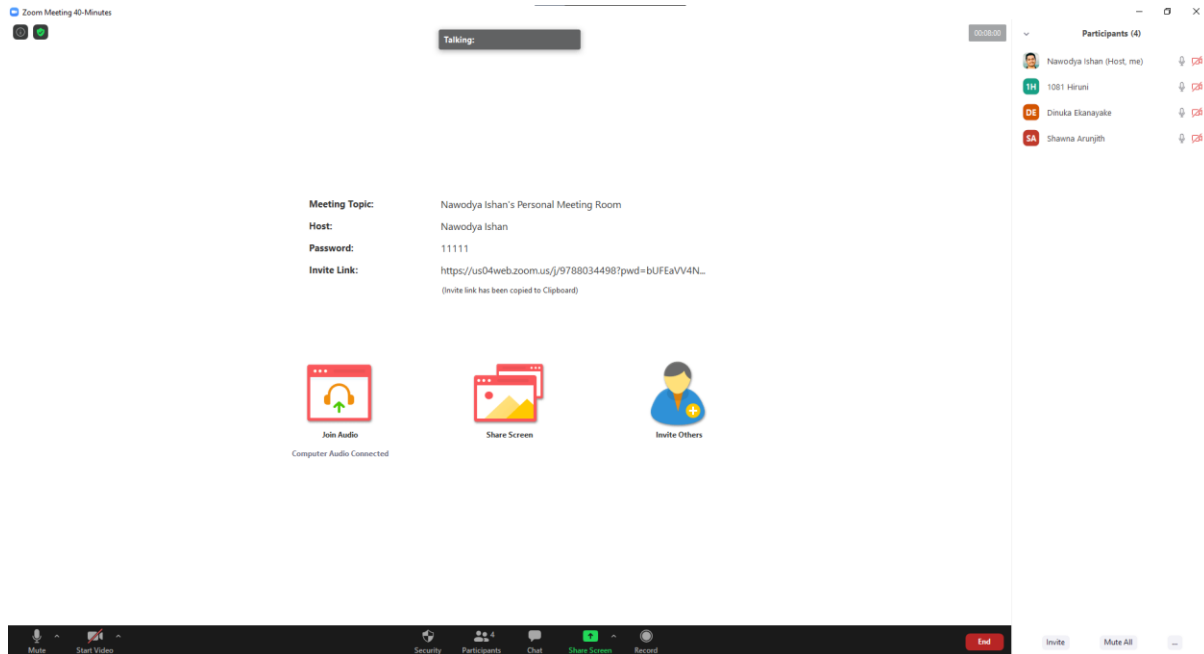
Attendance

Meeting Date	Meeting Platform	Meeting Objectives	Attendance
2020.03.15	WhatsApp	Discussion about project initial planning	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.03.25	WhatsApp	Discussion about making project proposal	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.04.04	Zoom Google Docs	Making project proposal through online google docs	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.04.06	WhatsApp	Resubmission of Proposal	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.04.07	Zoom Google Docs	Finalizing Project Proposal	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.04.25	WhatsApp	Discussion about project SRS document	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.05.01	Zoom Google Docs	Making Project SRS Document	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.05.09	Zoom Google Docs	Finalizing SRS Document	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.05.10	WhatsApp	Discussion about project Interim Report	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.05.17	Zoom Google Docs	Finalizing Interim Report Document	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.05.25	Zoom	Discussion and making of Login Management System	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya

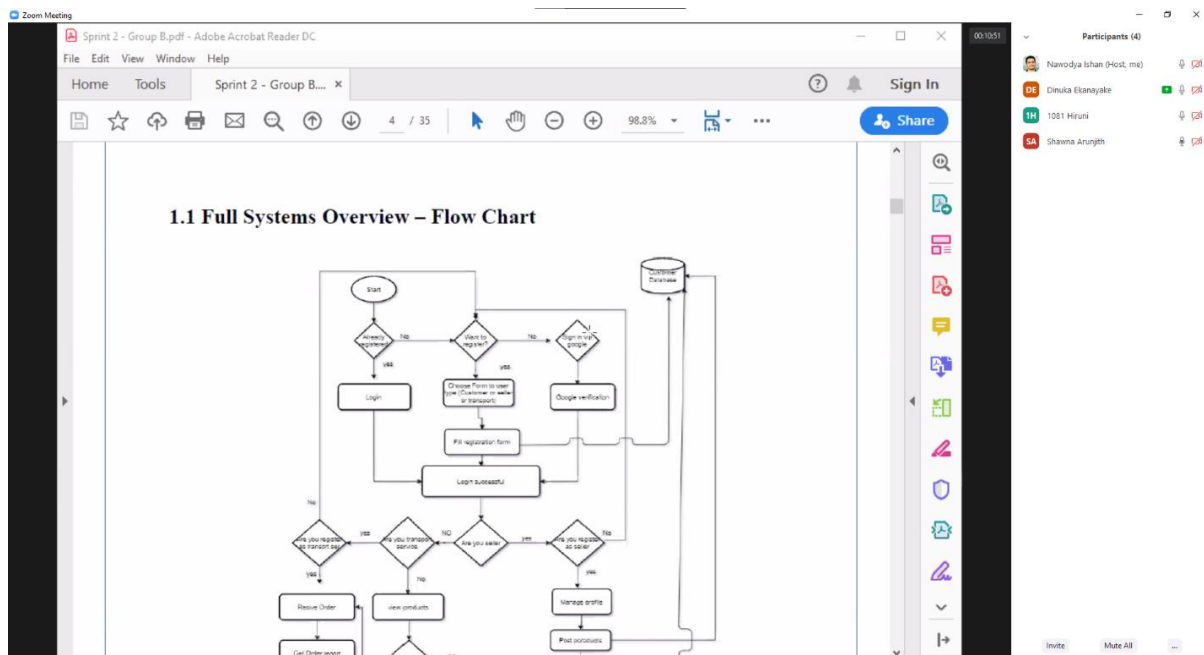
2020.06.01	Zoom	Testing Login management System	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.06.04	Zoom	Finalizing Login management system and submission of Sprint 1	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.06.10	WhatsApp Google Docs	Making of Sprint 2 Backlog	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.06.18	Zoom	Testing Sprint 2 with more sub systems	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.05.25	Zoom	Finalizing Sprint 2 System Release	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.07.23	Zoom	Discussion about submission of Individual reports and Sprint 3	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.07.27	WhatsApp	Testing Sprint 3	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.08.01	Zoom	Discussion about further improvements	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya
2020.08.08	Zoom WhatsApp Google Docs	Finalizing Individual Reports and Sprint 3	Dinuka Ekanayake Shawn Arunjith Nawodya Ishan Tharushika Balasooriya

Evidence

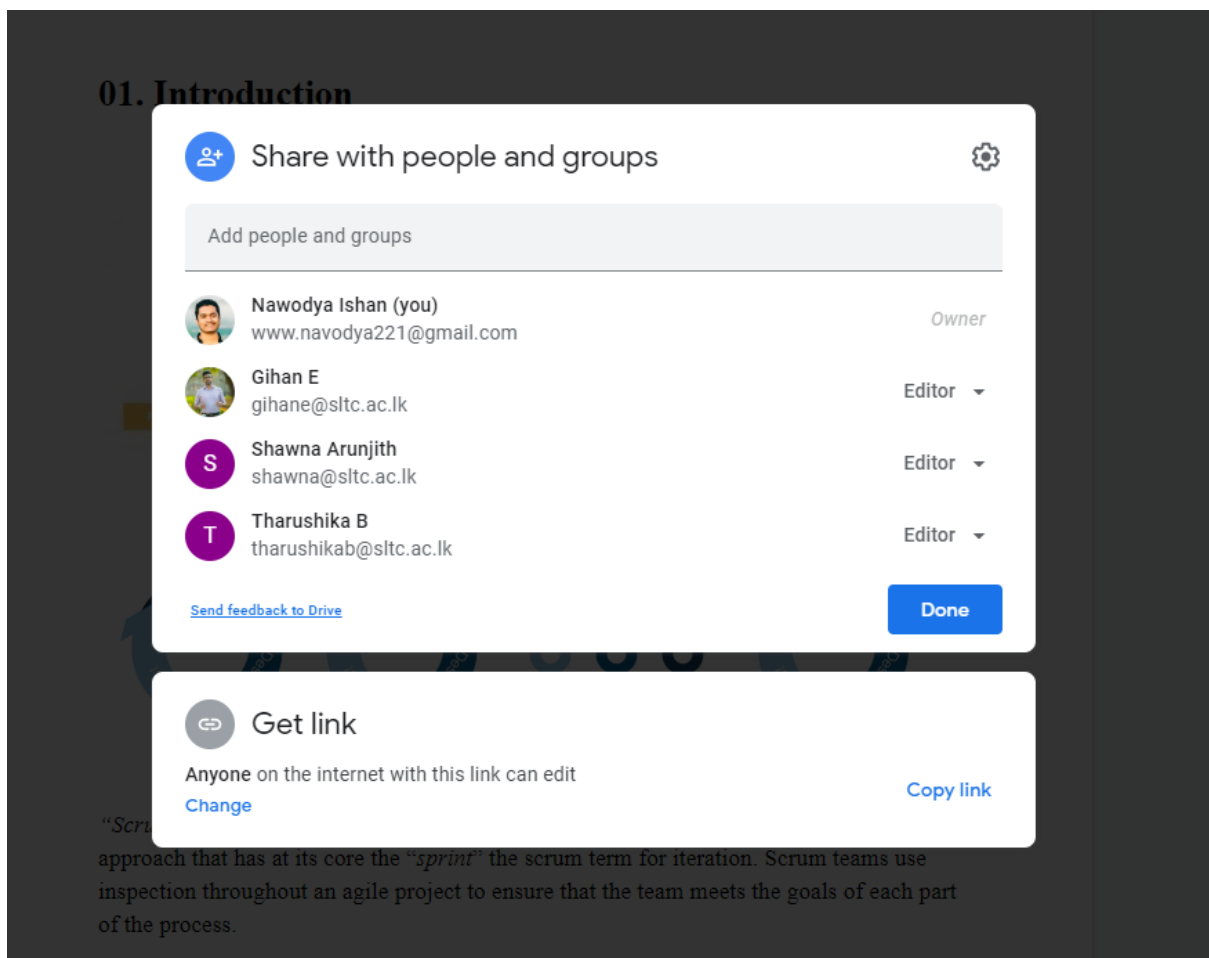
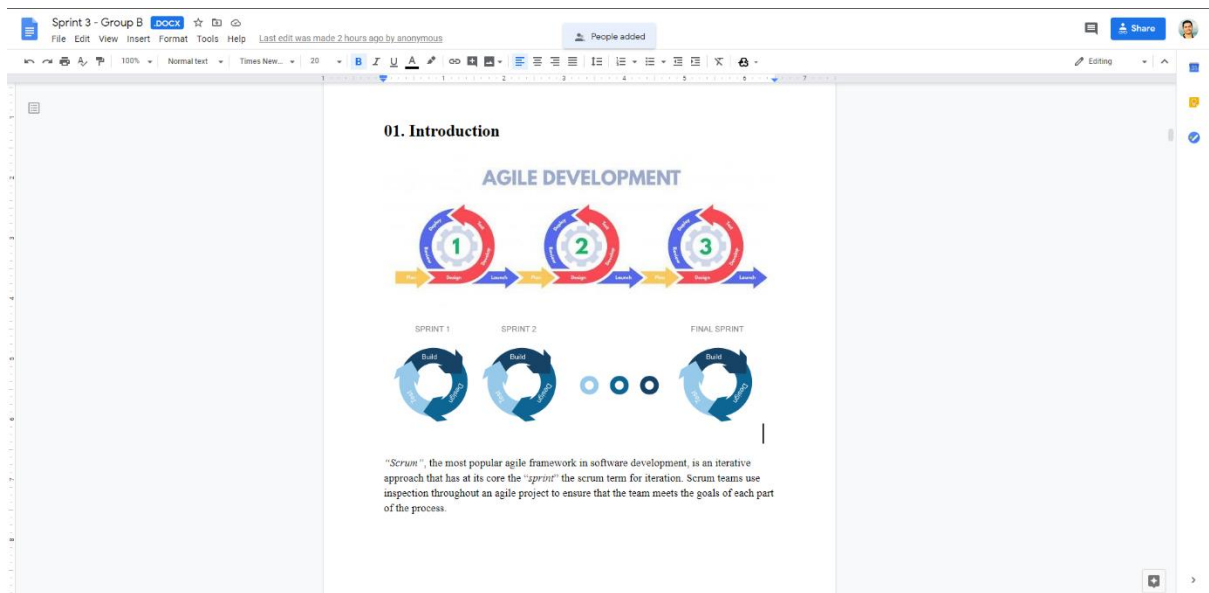
1) Zoom Audio Discussion



2) Zoom Video Discussion



3) Google Docs



4) WhatsApp Group Communication



Choosing the relevant technologies

When choosing technologies, we could implement this project as android app or website. And we choose to implement as android app because nowadays everyone has smart phones and app is very easy to use and comfortable to user than a website.

For create android app, basically we use android studio software for development and google firebase for manage database. In android studio we use java language to code our app. Java language is easy to use. It was designed to be easy to use, write, compile, debug, and learn than other programming languages. And java is an object-oriented language. It associated with concepts like class, object, inheritance, encapsulation, abstraction, polymorphism, etc. which allows us to create modular programs and reusable code. We can declare classes, create objects inside classes, and interact between two objects.

We used firebase to manage our database. It gives functionality like analytics, databases, messaging and crash reporting. Firebase is built on Google infrastructure and scales automatically.

Consider about disadvantages of technology that we choose; it basically gets problems with performance because android studio and java language significantly slower and more memory-consuming.

Implementing functional requirements

- Login system

This is a code for creating and describing variables, buttons, login as admin option and reset password option.

```
1 package lk.slrc.agrimarket;
2
3 import ...
4
26
27 public class LoginActivity extends AppCompatActivity {
28     private EditText InputMobileNo, InputPassword;
29     private Button Login;
30     private ProgressDialog loadingBar;
31     private TextView AdminLink, NotAdminLink;
32
33     private String parentDbName = "Users";
34     private CheckBox chkBoxRememberMe;
35     private TextView forgotPassword;
36
37     @Override
38     protected void onCreate(Bundle savedInstanceState) {
39         super.onCreate(savedInstanceState);
40         setContentView(R.layout.activity_login);
41
42         Login = (Button) findViewById(R.id.btn_Login);
43         InputMobileNo = (EditText) findViewById(R.id.Login_mobile);
44         InputPassword = (EditText) findViewById(R.id.Login_password);
45         AdminLink = (TextView) findViewById(R.id.admin_panel);
46         NotAdminLink = (TextView) findViewById(R.id.not_admin_panel);
47         loadingBar = new ProgressDialog( context: this);
48         forgotPassword = (TextView) findViewById(R.id.frgt_pw);
49         chkBoxRememberMe = (CheckBox) findViewById(R.id.chkbb_stay);
50         Paper.init( context: this);
51     }
52 }
```



```

52 Login.setOnClickListener(new View.OnClickListener() {
53     @Override
54     public void onClick(View v) {
55         LoginUser();
56     }
57 });
58
59 AdminLink.setOnClickListener(new View.OnClickListener() {           //When user wants to login as admin
60     @Override
61     public void onClick(View view) {
62         Login.setText("Admin Login");
63         AdminLink.setVisibility(View.INVISIBLE);
64         NotAdminLink.setVisibility(View.VISIBLE);
65         chkBoxRememberMe.setVisibility(View.INVISIBLE);
66         forgotPassword.setVisibility(View.INVISIBLE);
67         parentDbName = "Admins";
68     }
69 });
70
71 NotAdminLink.setOnClickListener(new View.OnClickListener() {           //Return to the user login
72     @Override
73     public void onClick(View view) {
74         Login.setText("Login");
75         AdminLink.setVisibility(View.VISIBLE);
76         NotAdminLink.setVisibility(View.INVISIBLE);
77         chkBoxRememberMe.setVisibility(View.VISIBLE);
78         forgotPassword.setVisibility(View.VISIBLE);
79         parentDbName = "Users";
80     }
81 });
82
83 forgotPassword.setOnClickListener(new View.OnClickListener() {
84     @Override
85     public void onClick(View view)
86     {
87         Intent intent = new Intent( packageContext: LoginActivity.this, ResetPasswordActivity.class);
88         intent.putExtra( name: "check", value: "login");
89         startActivity(intent);
90     }
91 });
92 }

```

This code running after user press login button. First it takes data that user added to mobile number and password into the string variables, then check the both fields are empty or not and if it is not empty display the loading bar and call function AllowAccessToAccount.

```

93
94 private void LoginUser()
95 {
96     String mobile = InputMobileNo.getText().toString();
97     String password = InputPassword.getText().toString();
98
99     if (TextUtils.isEmpty(mobile))
100     {
101         Toast.makeText( context: this, text: "Please enter your Mobile no.", Toast.LENGTH_SHORT).show();
102     }
103     else if (TextUtils.isEmpty(password))
104     {
105         Toast.makeText( context: this, text: "Please enter your Password", Toast.LENGTH_SHORT).show();
106     }
107     else {
108         loadingBar.setTitle("Login Account");
109         loadingBar.setMessage("Please wait...");
110         loadingBar.setCanceledOnTouchOutside(false);
111         loadingBar.show();
112
113         AllowAccessToAccount(mobile,password);
114     }
115 }
116

```

In AllowAccessToAccount function, verify user data by checking data in the database.

```
117 private void AllowAccessToAccount(final String mobile, final String password) {
118
119     if(chkBoxRememberMe.isChecked()){
120         Paper.book().write(Prevalent.UserPhoneKey, mobile);
121         Paper.book().write(Prevalent.UserPasswordKey, password);
122     }
123
124     final DatabaseReference RootRef;
125     RootRef = FirebaseDatabase.getInstance().getReference();
126
127     RootRef.addListenerForSingleValueEvent(new ValueEventListener() //Checking data that user input for login
128     {
129         @Override
130         public void onDataChange(@NonNull DataSnapshot dataSnapshot)
131         {
132             if(dataSnapshot.child(parentDbName).child(mobile).exists())
133             {
134                 Users usersData = dataSnapshot.child(parentDbName).child(mobile).getValue(Users.class);
135
136                 if (usersData.getMobile().equals(mobile))
137                 {
138                     if (usersData.getPassword().equals(password))
139                     {
140                         if (parentDbName.equals("Admins")) //When trying to login as admin
141                         {
142                             Toast.makeText( context: LoginActivity.this, text: "Logged in as Admin Successfully", Toast.LENGTH_SHORT).show();
143                             loadingBar.dismiss();
144
145                             Intent intent = new Intent( packageContext: LoginActivity.this, AdminHomeActivity.class);
146                             startActivity(intent);
147                         }
148                         else if (parentDbName.equals("Users")) //When trying to login as user
149                         {
150                             Toast.makeText( context: LoginActivity.this, text: "Logged in Successfully", Toast.LENGTH_SHORT).show();
151                             loadingBar.dismiss();
152
153                             Intent intent = new Intent( packageContext: LoginActivity.this, HomeActivity.class);
154                             Prevalent.currentOnlineUser = usersData;
155                             startActivity(intent);
156                         }
157                     }
158                     else {
159                         loadingBar.dismiss();
160                         Toast.makeText( context: LoginActivity.this, text: "Incorrect Password", Toast.LENGTH_SHORT).show();
161                     }
162                 }
163             }
164             else {
165                 Toast.makeText( context: LoginActivity.this, text: "Incorrect Username", Toast.LENGTH_SHORT).show();
166                 loadingBar.dismiss();
167             }
168         }
169     });
170
171     @Override
172     public void onCancelled(@NonNull DatabaseError databaseError) {
```

• Register

Here we create variables, describe them and call function CreateAccount when user press the create account button.

```
25 public class RegisterActivity extends AppCompatActivity {
26     private Button CreateAccount;
27     private EditText InputName, InputMobileNo, InputEmail, InputPassword, InputAddress;
28     private ProgressDialog loadingBar;
29     private String emailPattern = "^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-Za-z0-9]+(\\.[A-Za-z0-9]+)*((\\.[A-Za-z]{2,})$)";
30
31     @Override
32     protected void onCreate(Bundle savedInstanceState) {
33         super.onCreate(savedInstanceState);
34         setContentView(R.layout.activity_register);
35
36         CreateAccount = (Button) findViewById(R.id.btn_create_account);
37         InputName = (EditText) findViewById(R.id.reg_name);
38         InputMobileNo = (EditText) findViewById(R.id.reg_mobile);
39         InputEmail = (EditText) findViewById(R.id.reg_email);
40         InputPassword = (EditText) findViewById(R.id.reg_password);
41         InputAddress = (EditText) findViewById(R.id.reg_address);
42         loadingBar = new ProgressDialog( context: this);
43
44         CreateAccount.setOnClickListener(new View.OnClickListener() {
45             @Override
46             public void onClick(View v)
47             {
48                 CreateAccount();
49             }
50         });
51     }
```

In `CreateAccount` function first we take data that input by user to string variables and check the validity of input data. If all data validated, it displays loading bar and call `ValidatMobileNumber` function.

```
53 private void CreateAccount()
54 {
55     String name = InputName.getText().toString();
56     String mobile = InputMobileNo.getText().toString();
57     String email = InputEmail.getText().toString().trim();
58     String password = InputPassword.getText().toString();
59     String address = InputAddress.getText().toString();
60
61     if (TextUtils.isEmpty(name))
62     {
63         Toast.makeText( context: this, text: "Please enter your Name", Toast.LENGTH_SHORT).show();
64     }
65     else if (TextUtils.isEmpty(mobile) || !(mobile.length() == 10) )
66     {
67         Toast.makeText( context: this, text: "Please enter valid Mobile no.", Toast.LENGTH_SHORT).show();
68     }
69     else if (TextUtils.isEmpty(email) || !email.matches(emailPattern))
70     {
71         Toast.makeText( context: this, text: "Please enter valid Email", Toast.LENGTH_SHORT).show();
72     }
73     else if (TextUtils.isEmpty(password) || password.length() < 6 )
74     {
75         Toast.makeText( context: this, text: "Your Password must have minimum 6 characters", Toast.LENGTH_SHORT).show();
76     }
77     else if (TextUtils.isEmpty(address))
78     {
79         Toast.makeText( context: this, text: "Please enter your Address", Toast.LENGTH_SHORT).show();
80     }
81     else
82     {
83         loadingBar.setTitle("Create Account");
84         loadingBar.setMessage("Please wait...");
85         loadingBar.setCanceledOnTouchOutside(false);
86         loadingBar.show();
87
88         ValidatMobileNumber(name, mobile, email, password, address);
89     }
90 }
```

In ValidatMobileNumber function it check the mobile number that input by user, is already exist in database. If it is not existing input user data to database and create new account. (We are checking about mobile number because we use mobile number as a primary key in database)

```
92 private void ValidatMobileNumber(final String name, final String mobile, final String email, final String password, final String address)
93 {
94     final DatabaseReference RootRef;
95     RootRef = FirebaseDatabase.getInstance().getReference();
96
97     RootRef.addListenerForSingleValueEvent(new ValueEventListener() {
98         @Override
99         public void onDataChange(@NonNull DataSnapshot dataSnapshot)
100         {
101             if (!(dataSnapshot.child("Users").child(mobile).exists()))
102             {
103                 HashMap<String, Object> userdataMap = new HashMap<>();
104                 userdataMap.put("mobile", mobile);
105                 userdataMap.put("name", name);
106                 userdataMap.put("email", email);
107                 userdataMap.put("password", password);
108                 userdataMap.put("address", address);
109
110                 RootRef.child("Users").child(mobile).updateChildren(userdataMap).addOnCompleteListener((task) -> {
111                     if (task.isSuccessful()) {
112                         Toast.makeText( context: RegisterActivity.this, text: "Congratulations! Your account created successfully!",
113                             loadingBar.dismiss();
114
115                         Intent intent = new Intent( packageContext: RegisterActivity.this, LoginActivity.class);
116                         startActivity(intent);
117                     }
118                     else{
119                         loadingBar.dismiss();
120                         Toast.makeText( context: RegisterActivity.this, text: "Error, Please Try Again", Toast.LENGTH_SHORT).show();
121                     }
122                 });
123             }
124         }
125     });
126
127     else {
128         Toast.makeText( context: RegisterActivity.this, text: "This number is in use", Toast.LENGTH_SHORT).show();
129         loadingBar.dismiss();
130
131         Intent intent = new Intent( packageContext: RegisterActivity.this, MainActivity.class);
132         startActivity(intent);
133     }
134 }
135
136
137 @Override
138 public void onCancelled(@NonNull DatabaseError databaseError) {
139
140 }
141 }
```

- **Cart**

Cart is main requirement for users. Users can add product to cart and when they want they can buy them or remove them from cart.

```
30 public class CartActivity extends AppCompatActivity {
31
32     private RecyclerView recyclerView;
33     private RecyclerView.LayoutManager layoutManager;
34     private Button NextBtn, viewTotal;
35     private TextView txtTotal;
36     private int totalPrice=0;
37
38     @Override
39     protected void onCreate(Bundle savedInstanceState) {
40         super.onCreate(savedInstanceState);
41         setContentView(R.layout.activity_cart);
42
43         recyclerView = findViewById(R.id.cart_list);
44         recyclerView.setHasFixedSize(true);
45         layoutManager = new LinearLayoutManager( context: this);
46         recyclerView.setLayoutManager(layoutManager);
47
48         NextBtn = (Button) findViewById(R.id.cart_next);
49         viewTotal = (Button) findViewById(R.id.cart_total);
50         txtTotal = (TextView) findViewById(R.id.cart_text_price);
51
52         NextBtn.setOnClickListener((v) → {
53             Intent intent = new Intent( packageContext: CartActivity.this, ConfirmFinalOrderActivity.class);
54             intent.putExtra( name: "Total Price", String.valueOf(totalPrice));
55             startActivity(intent);
56             finish();
57         });
58
59         viewTotal.setOnClickListener((v) → {
60             txtTotal.setText("Total Price = Rs." + String.valueOf(totalPrice));
61             Intent intent = new Intent( packageContext: CartActivity.this, CartActivity.class);
62             intent.putExtra( name: "Total Price", String.valueOf(totalPrice));
63         });
64     }
65 }
```

```

72  @Override
73  protected void onStart() {
74      super.onStart();
75
76      final DatabaseReference cartListRef = FirebaseDatabase.getInstance().getReference().child("Cart List");
77
78      FirebaseRecyclerOptions<Cart> options = new FirebaseRecyclerOptions.Builder<Cart>().setQuery(cartListRef.child("User View")
79          .child(Prevalent.currentOnlineUser.getMobile()).child("Products"), Cart.class).build();
80
81      FirebaseRecyclerAdapter<Cart, CartViewHolder> adapter = new FirebaseRecyclerAdapter<>>(options) {
82          @Override
83          protected void onBindViewHolder(@NonNull CartViewHolder holder, int position, @NonNull final Cart model) {
84              holder.txtProductQuantity.setText("Quantity = " + model.getQuantity());
85              holder.txtProductPrice.setText("Price = Rs." + model.getPrice());
86              holder.txtProductName.setText(model.getName());
87
88              int oneProductTotalPrice = ((Integer.valueOf(model.getPrice()))) * Integer.valueOf(model.getQuantity());
89              totalPrice = totalPrice + oneProductTotalPrice;
90
91              holder.itemView.setOnClickListener((v) -> { //Let user to edit product in cart or remove it from cart
92                  CharSequence options[] = new CharSequence[]{
93                      "Edit",
94                      "Remove"
95                  };
96                  AlertDialog.Builder builder = new AlertDialog.Builder( context: CartActivity.this);
97                  builder.setTitle("Cart Options:");
98
99                  builder.setItems(options, (dialogInterface, i) -> {
100                      if (i == 0){
101                          Intent intent = new Intent( packageContext: CartActivity.this, ProductDetailsActivity.class);
102                          intent.putExtra( name: "pid", model.getPid());
103                          startActivity(intent);
104                      }
105                      if (i == 1) {
106                          cartListRef.child("User View")
107                              .child(Prevalent.currentOnlineUser.getMobile())
108                              .child("Products")
109                              .child(model.getPid())
110                              .removeValue()
111                              .addOnCompleteListener((task) -> {
112                                  if (task.isSuccessful()){
113                                      Toast.makeText( context: CartActivity.this, text: "Item removed successfully", Toast.LENGTH
114                                          );
115                                  }
116                              });
117                      }
118                  });
119                  builder.show();
120              });
121          }
122      };
123
124      @NonNull
125      @Override
126      public CartViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
127          View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.cart_item_layout, parent, attachToRoot: false);
128          CartViewHolder holder = new CartViewHolder(view);
129          return holder;
130      }
131
132      };
133
134      recyclerView.setAdapter(adapter);
135      adapter.startListening();
136  }
137  }
138  }
139  }
140  }
141  }
142  }
143  }

```

- **Home**

Browsing for products is main function of this app. We use recycle view for show products to user. User can select product and view details of product and add it to cart.

```
25 public class Home2Activity extends AppCompatActivity {
26     private ImageView Back;
27     private DatabaseReference ProductRef;
28     private RecyclerView recyclerView;
29     RecyclerView.LayoutManager layoutManager;
30
31     private String type = "";
32
33     @Override
34     protected void onCreate(Bundle savedInstanceState) {
35         super.onCreate(savedInstanceState);
36         setContentView(R.layout.activity_home2);
37
38         Intent intent = getIntent();
39         Bundle bundle = intent.getExtras();
40         if (bundle != null)
41         {
42             type = getIntent().getExtras().get("Admin").toString();
43         }
44
45         ProductRef = FirebaseDatabase.getInstance().getReference().child("Products");
46
47         Back = (ImageView) findViewById(R.id.home2_back);
48         recyclerView = findViewById(R.id.recycler_menu);
49         recyclerView.setHasFixedSize(true);
50         layoutManager = new LinearLayoutManager( context: this);
51         recyclerView.setLayoutManager(layoutManager);
52
53         if (type.equals("Admin")){
54             Back.setOnClickListener((v) -> {
55                 Intent intent = new Intent( packageContext: Home2Activity.this, AdminHomeActivity.class);
56                 startActivity(intent);
57             });
58         }
59         else {
60             Back.setOnClickListener((v) -> {
61                 Intent intent = new Intent( packageContext: Home2Activity.this, HomeActivity.class);
62                 startActivity(intent);
63             });
64         }
65     }
66 }
67
68
69
70
71 }
```



```

73      @Override
74      protected void onStart() {
75          super.onStart();
76
77          FirebaseRecyclerOptions<Products> options =
78              new FirebaseRecyclerOptions.Builder<Products>().setQuery(ProductRef.orderByChild("productState")
79                  .equalTo("Approved"), Products.class).build();
80
81          FirebaseRecyclerAdapter<Products, ProductViewHolder> adapter = new FirebaseRecyclerAdapter<>>(options) {
82              @Override
83              protected void onBindViewHolder(@NonNull ProductViewHolder holder, int position, @NonNull final Products model) {
84                  holder.txtProductName.setText(model.getName());
85                  holder.txtProductDescription.setText(model.getDescription());
86                  holder.txtProductPrice.setText("Price = Rs." + model.getPrice());
87                  Picasso.get().load(model.getImage()).into(holder.imageView);
88
89                  holder.itemView.setOnClickListener((v) -> {
90                      if (type.equals("Admin")){
91                          Intent intent = new Intent( packageContext: Home2Activity.this, AdminMaintainProductsActivity.class);
92                          intent.putExtra( name: "pid", model.getPid());
93                          startActivity(intent);
94                      }
95                      else{
96                          Intent intent = new Intent( packageContext: Home2Activity.this, ProductDetailsActivity.class);
97                          intent.putExtra( name: "pid", model.getPid());
98                          startActivity(intent);
99                      }
100                  });
101
102              @NonNull
103              @Override
104              public ProductViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
105                  View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.product_items_layout, parent, attachToRoot: false)
106                  ProductViewHolder holder = new ProductViewHolder(view);
107                  return holder;
108              }
109          };
110
111          recyclerView.setAdapter(adapter);
112          adapter.startListening();
113      }
114  }
115
116  }
117
118  }
119

```

• Add new product

Only sellers can add new products to app sellers must create account and after that they can add new products, that products view in users' home after admins approve. After adding, sellers can remove or change details of their products any time.

```
39 public class SellersAddNewProductActivity extends AppCompatActivity {
40     private String CategoryName, Description, Price, PName, saveCurrentDate, saveCurrentTime;
41     private Button AddNewProduct;
42     private ImageView InputProductImage;
43     private EditText InputProductName, InputProductDescription, InputProductPrice;
44     private static final int GalleryPick = 1;
45     private Uri ImageUri;
46     private String ProductRandomKey, downloadImageUrl;
47     private StorageReference ProductImageRef;
48     private DatabaseReference ProductsRef, sellersRef;
49     private ProgressDialog loadingBar;
50
51     private String sName, sAddress, sMobile;
52
53     @Override
54     protected void onCreate(Bundle savedInstanceState) {
55         super.onCreate(savedInstanceState);
56         setContentView(R.layout.activity_sellers_add_new_product);
57
58         CategoryName = getIntent().getExtras().get("category").toString();
59         ProductImageRef = FirebaseStorage.getInstance().getReference().child("Product Images");
60         ProductsRef = FirebaseDatabase.getInstance().getReference().child("Products");
61         sellersRef = FirebaseDatabase.getInstance().getReference().child("Sellers");
62
63         AddNewProduct = (Button) findViewById(R.id.add_new_product);
64         InputProductImage = (ImageView) findViewById(R.id.select_product_image);
65         InputProductName = (EditText) findViewById(R.id.product_name);
66         InputProductDescription = (EditText) findViewById(R.id.product_description);
67         InputProductPrice = (EditText) findViewById(R.id.product_price);
68         loadingBar = new ProgressDialog(context, this);
69
70         InputProductImage.setOnClickListener((v) -> { OpenGallery(); });
71
72         AddNewProduct.setOnClickListener((v) -> { ValidateProductData(); });
73
74         sellersRef.child(Prevalent2.currentOnlineUser.getMobile()).addValueEventListener(new ValueEventListener() {
75             @Override
76             public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
77                 if (dataSnapshot.exists()){
78                     sName = dataSnapshot.child("name").getValue().toString();
79                     sAddress = dataSnapshot.child("address").getValue().toString();
80                     sMobile = dataSnapshot.child("mobile").getValue().toString();
81                 }
82             }
83
84             @Override
85             public void onCancelled(@NonNull DatabaseError databaseError) {
86
87             }
88         });
89     }
90 }
```

```

101 private void OpenGallery() { //adding product image
102     Intent galleryIntent = new Intent();
103     galleryIntent.setAction(Intent.ACTION_GET_CONTENT);
104     galleryIntent.setType("image/*");
105     startActivityForResult(galleryIntent, GalleryPick);
106 }
107
108 @Override
109 protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
110     super.onActivityResult(requestCode, resultCode, data);
111
112     if (requestCode==GalleryPick && resultCode==RESULT_OK && data!=null){
113         ImageUri = data.getData();
114         InputProductImage.setImageURI(ImageUri);
115     }
116 }
117
118 private void ValidateProductData() {
119     Description = InputProductDescription.getText().toString();
120     Price = InputProductPrice.getText().toString();
121     PName = InputProductName.getText().toString();
122
123     if (ImageUri == null){
124         Toast.makeText( context: this, text: "Product image is mandatory", Toast.LENGTH_SHORT).show();
125     }
126     else if (TextUtils.isEmpty(Description)){
127         Toast.makeText( context: this, text: "Please write product description", Toast.LENGTH_SHORT).show();
128     }
129     else if (TextUtils.isEmpty(Price)){
130         Toast.makeText( context: this, text: "Please write product price", Toast.LENGTH_SHORT).show();
131     }
132     else if (TextUtils.isEmpty(PName)){
133         Toast.makeText( context: this, text: "Please write product Name", Toast.LENGTH_SHORT).show();
134     }
135     else {
136         StoreProductInformation();
137     }
138 }
139

```

```

141 private void StoreProductInformation() {
142     loadingBar.setTitle("Adding New Product");
143     loadingBar.setMessage("Please wait while we adding...");
144     loadingBar.setCanceledOnTouchOutside(false);
145     loadingBar.show();
146
147     Calendar calendar = Calendar.getInstance();
148
149     SimpleDateFormat currentDate = new SimpleDateFormat( pattern: "MM dd, yyyy");
150     saveCurrentDate = currentDate.format(calendar.getTime());
151
152     SimpleDateFormat currentTime = new SimpleDateFormat( pattern: "HH:mm:ss a");
153     saveCurrentTime = currentTime.format(calendar.getTime());
154
155     ProductRandomKey = saveCurrentDate + saveCurrentTime;
156
157
158     final StorageReference filePath = ProductImageRef.child(ImageUri.getLastPathSegment() + ProductRandomKey + ".jpg");
159
160     final UploadTask uploadTask = filePath.putFile(ImageUri);
161
162     uploadTask.addOnFailureListener((e) → {
163         String message = e.toString();
164         Toast.makeText( context: SellersAddNewProductActivity.this, text: "Error", Toast.LENGTH_SHORT).show();
165         loadingBar.dismiss();
166     }).addOnSuccessListener((OnSuccessListener) (taskSnapshot) → {
167         Toast.makeText( context: SellersAddNewProductActivity.this, text: "Image Uploaded Successfully", Toast.LENGTH_SHORT).show();
168
169         Task<Uri> urlTask = uploadTask.continueWithTask((task) → {
170             if (!task.isSuccessful()){
171                 throw task.getException();
172             }
173
174             downloadImageUrl = filePath.getDownloadUrl().toString();
175             return filePath.getDownloadUrl();
176         }).addOnCompleteListener((task) → {
177             if (task.isSuccessful()){
178                 downloadImageUrl = task.getResult().toString();
179                 Toast.makeText( context: SellersAddNewProductActivity.this, text: "Got the image URL successfully", Toast.LENGTH_SHORT).show();
180
181                 SaveProductInfoToDatabase();
182             }
183         });
184     });
185 }

```

```

201 private void SaveProductInfoToDatabase() {
202     HashMap<String, Object> productMap = new HashMap<>();
203     productMap.put("pid", ProductRandomKey);
204     productMap.put("date", saveCurrentDate);
205     productMap.put("time", saveCurrentTime);
206     productMap.put("description", Description);
207     productMap.put("image", downloadImageUrl);
208     productMap.put("category", CategoryName);
209     productMap.put("price", Price);
210     productMap.put("name", PName);
211
212     productMap.put("sellerName", sName);
213     productMap.put("sellerMobile", sMobile);
214     productMap.put("sellerAddress", sAddress);
215     productMap.put("productState", "Not Approved");
216
217     ProductsRef.child(ProductRandomKey).updateChildren(productMap)
218     .addOnCompleteListener((task) → {
219         if (task.isSuccessful()){
220             Intent intent =new Intent( packageContext: SellersAddNewProductActivity.this, SellerHomeActivity.class);
221             startActivity(intent);
222
223             loadingBar.dismiss();
224             Toast.makeText( context: SellersAddNewProductActivity.this, text: "Product added successfully", Toast.LENGTH_SHORT).show();
225
226         }
227         else{
228             loadingBar.dismiss();
229             String message= task.getException().toString();
230             Toast.makeText( context: SellersAddNewProductActivity.this, text: "Error" + message, Toast.LENGTH_SHORT).show();
231         }
232     });
233 }
234
235 }
236
237 }

```

Bug fixing

Here we consider about few bugs we found at sprint 2.

Mobile number and email format not verified:



A screenshot of a mobile application's "Enter Your Details" screen. The screen has a background image of a field at sunset. At the top, the title "Enter Your Details" is displayed. Below it are four input fields: a name field containing "Nawodya Ishan", a mobile number field containing "07410886649049041111", an email field containing "sakaibdkslahmslsjdmakssm", and a password field with masked characters. A red "Create Account" button is at the bottom. The status bar at the top shows the time as 8:17 PM and a data speed of 0.1KB/s.

Fixed: We set the length of mobile number to 10 and check whether email address is insert according to the correct pattern.

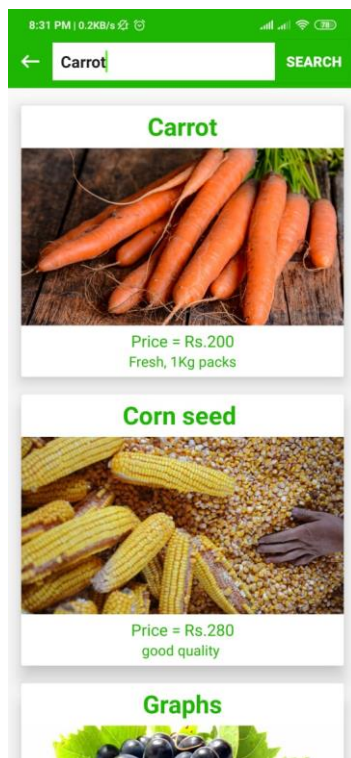


A screenshot of the "Enter Your Details" screen after a bug fix. The input fields now contain: "SLTC ICE 1" for the name, "0712200922" for the mobile number, "Slit" for the email, and "Sls sls sls" for the password. A red error message "Please enter valid Mobile no." is displayed below the mobile number field. The "Create Account" button remains at the bottom. The status bar shows the time as 8:36 PM and a data speed of 1.8KB/s.

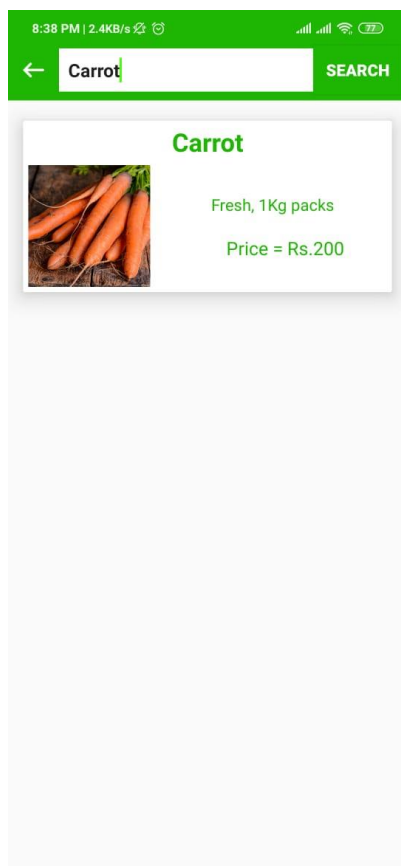


A screenshot of the "Enter Your Details" screen after a bug fix. The input fields now contain: "SLTC ICE 1" for the name, "0712200922" for the mobile number, "Slit" for the email, and "Sls sls sls" for the password. A red error message "Please enter valid Email" is displayed below the email field. The "Create Account" button remains at the bottom. The status bar shows the time as 8:36 PM and a data speed of 0.3KB/s.

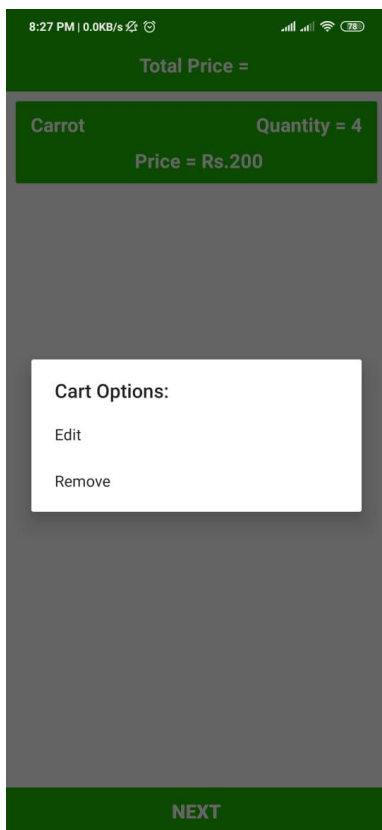
Search is not working correctly: It displays item that you search first and some other items.



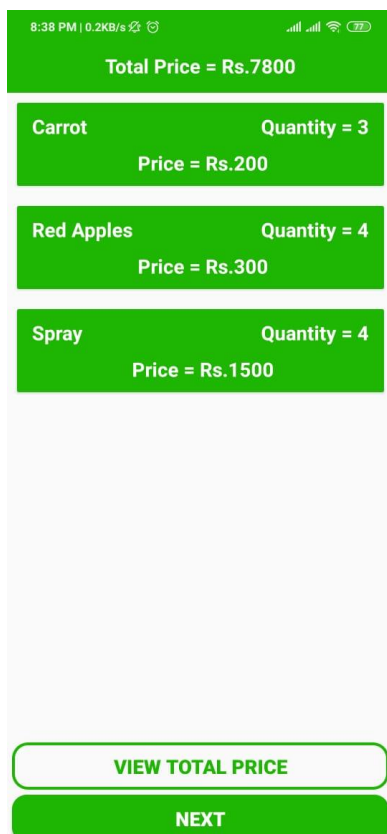
Fixed: Now it displays only item that you search.



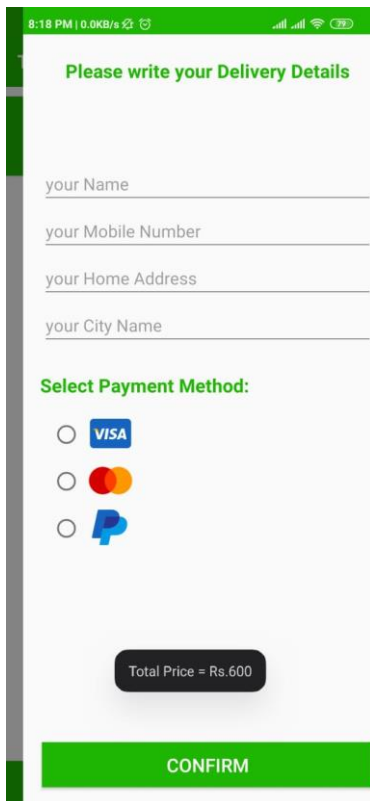
Total price of a cart not displayed:



Fixed: We add additional button to view total price.



User data didn't display at delivery details:



8:18 PM | 0.0KB/s

Please write your Delivery Details

your Name

your Mobile Number

your Home Address

your City Name

Select Payment Method:

☐ VISA

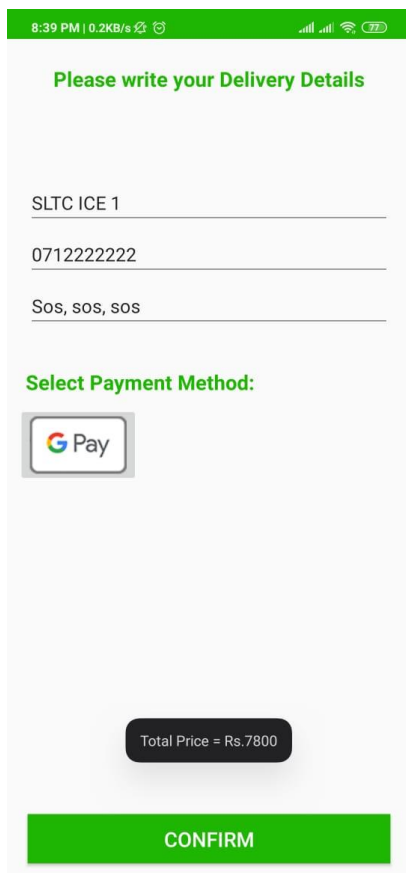
☐ Mastercard

☐ PayPal

Total Price = Rs.600

CONFIRM

Fixed:



8:39 PM | 0.2KB/s

Please write your Delivery Details

SLTC ICE 1

0712222222

Sos, sos, sos

Select Payment Method:

☒ G Pay


Total Price = Rs.7800

CONFIRM

User data didn't display at edit profile:

8:32 PM | 0.2KB/s

CloseUpdate



Change Image

Mobile Number

Name

Password


Address

SET SECURITY QUESTIONS

Fixed:

8:36 PM | 0.4KB/s

CloseUpdate



Change Image

0712222222

SLTC ICE 1

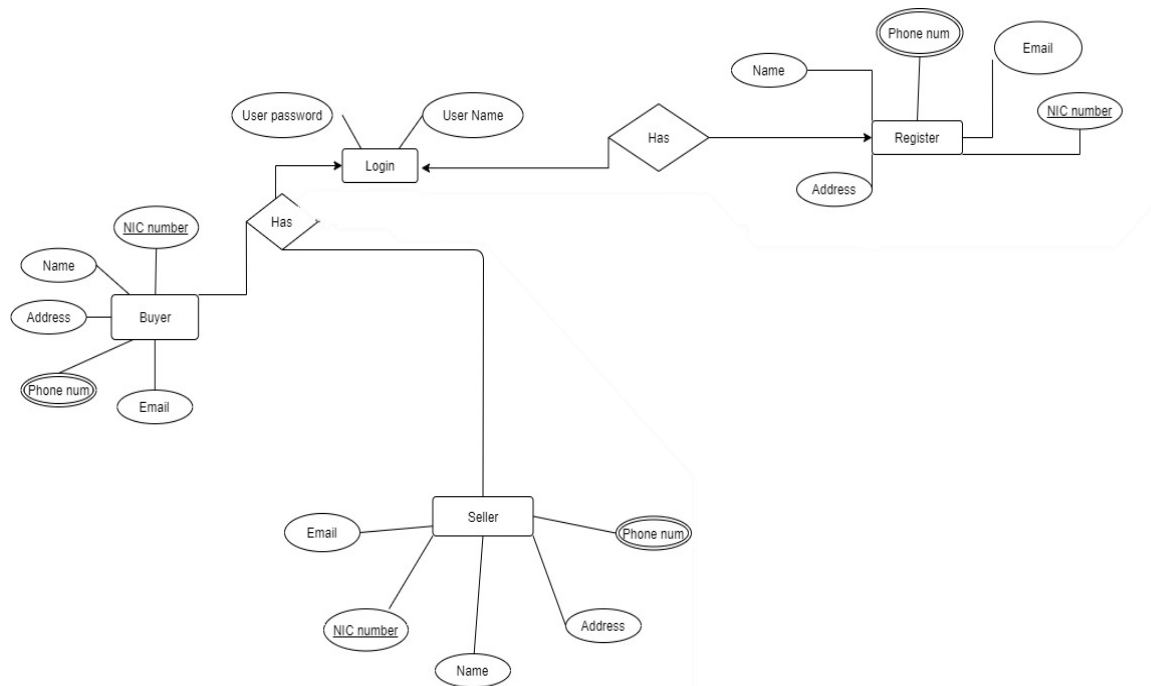
123456

Sltc@sos.com

Sos, sos, sos

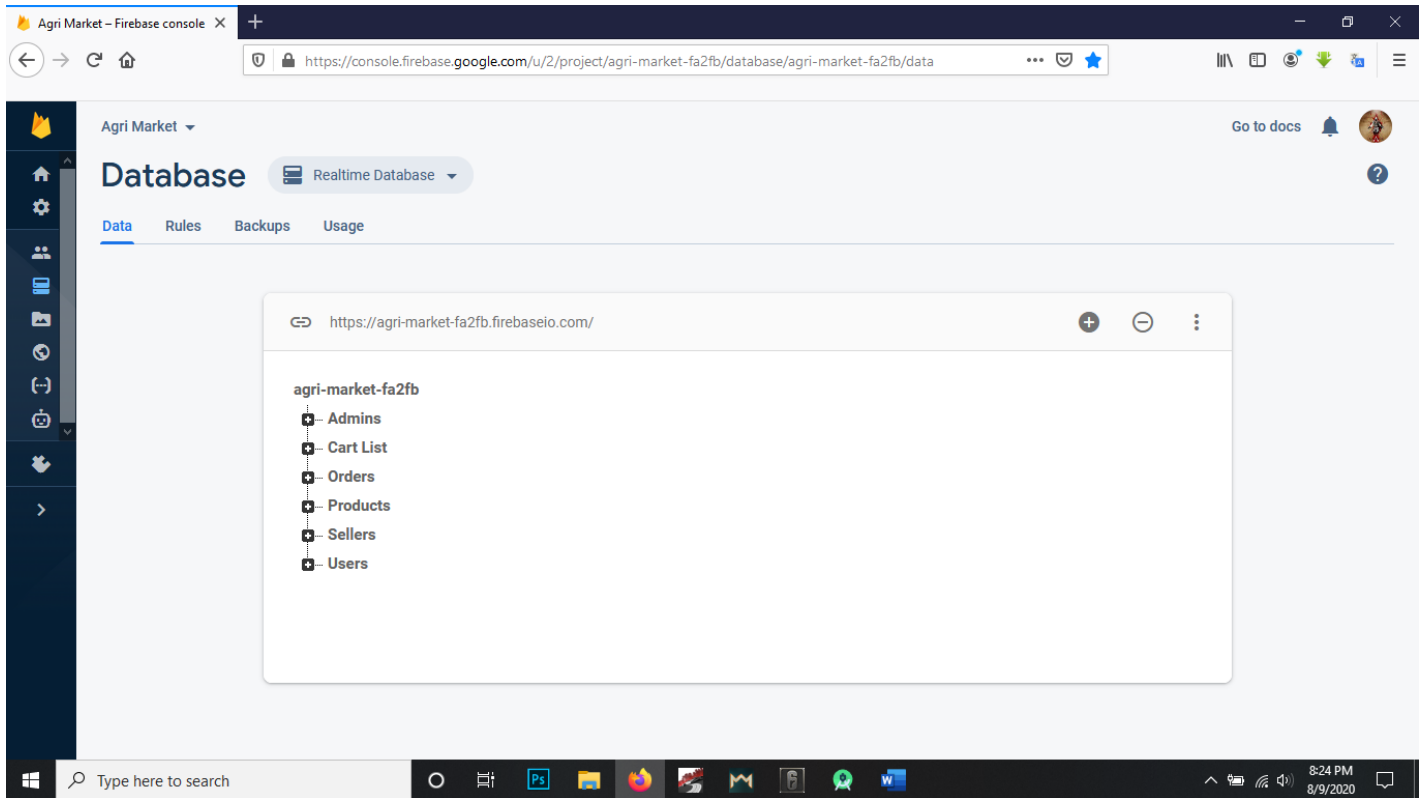
SET SECURITY QUESTIONS

Entity Relationship Diagram

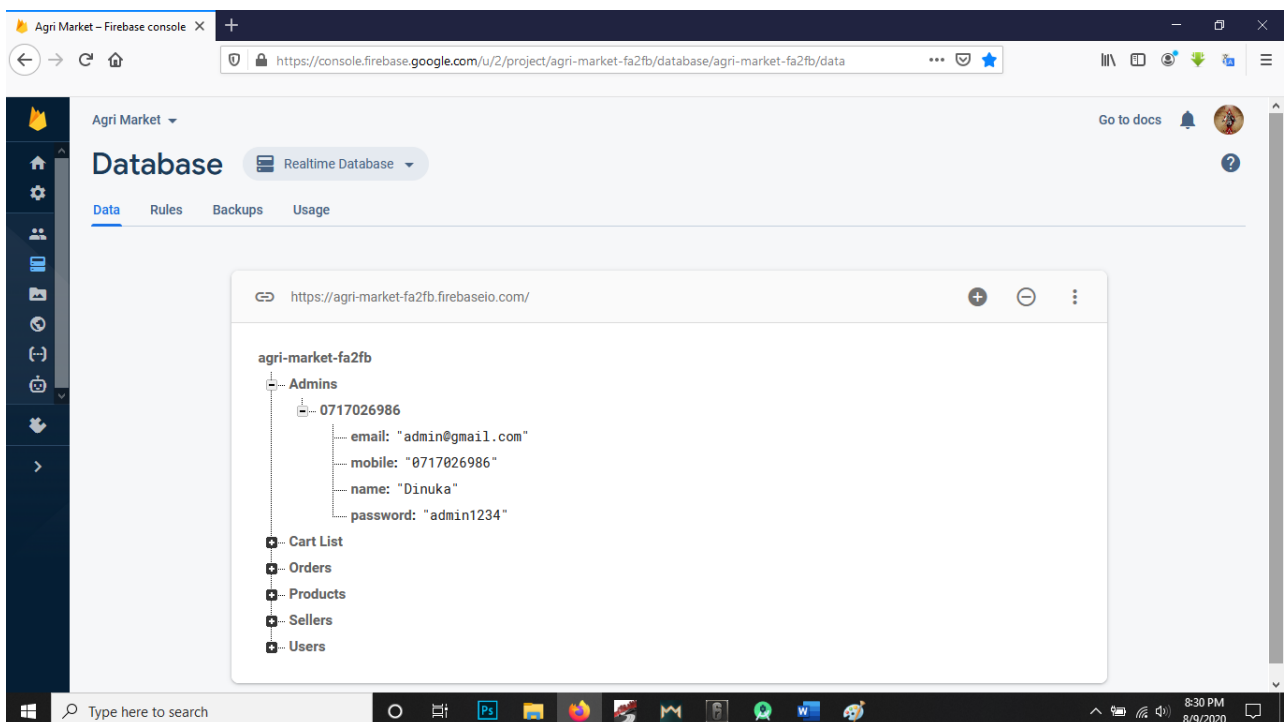


DDL Scripts

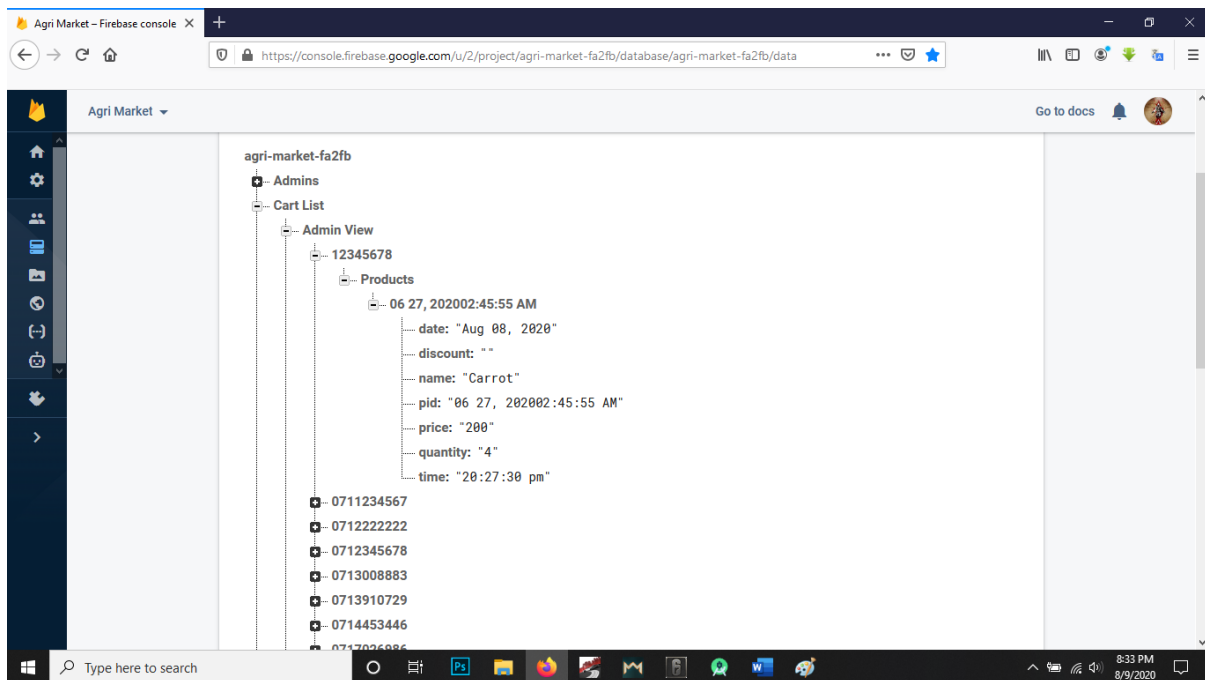
We use firebase console to manage our databases. There we use six different tables to represent admins, users, sellers, products, orders and cart list.



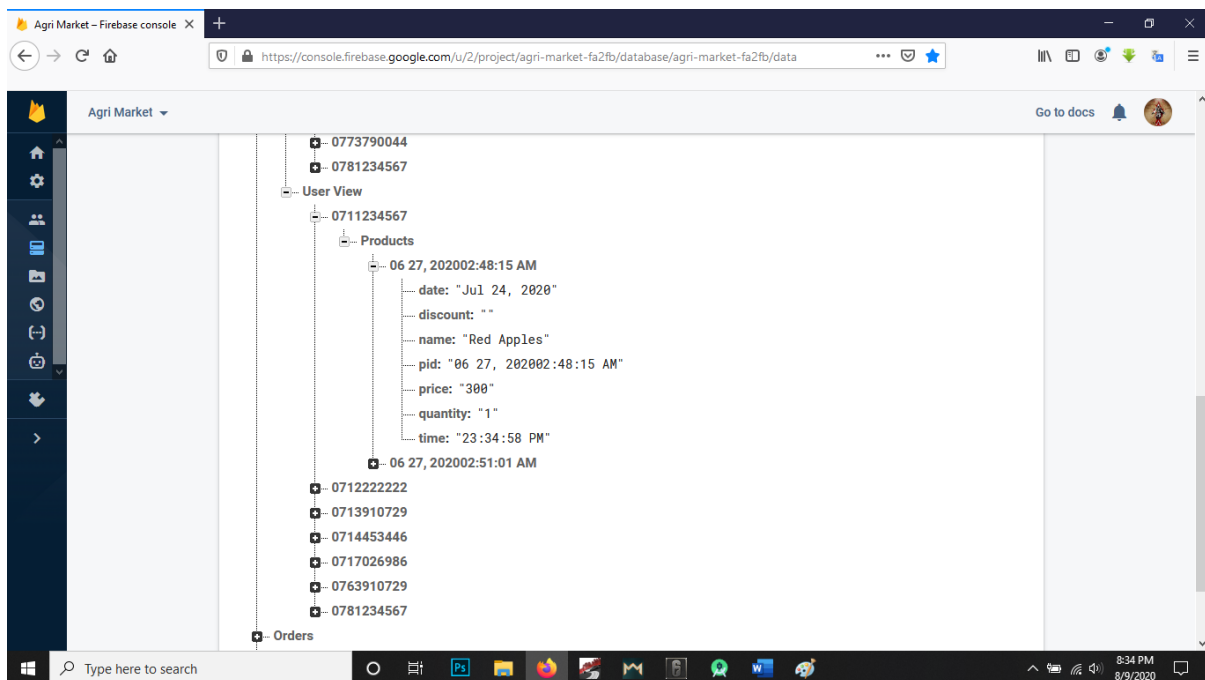
Admins table



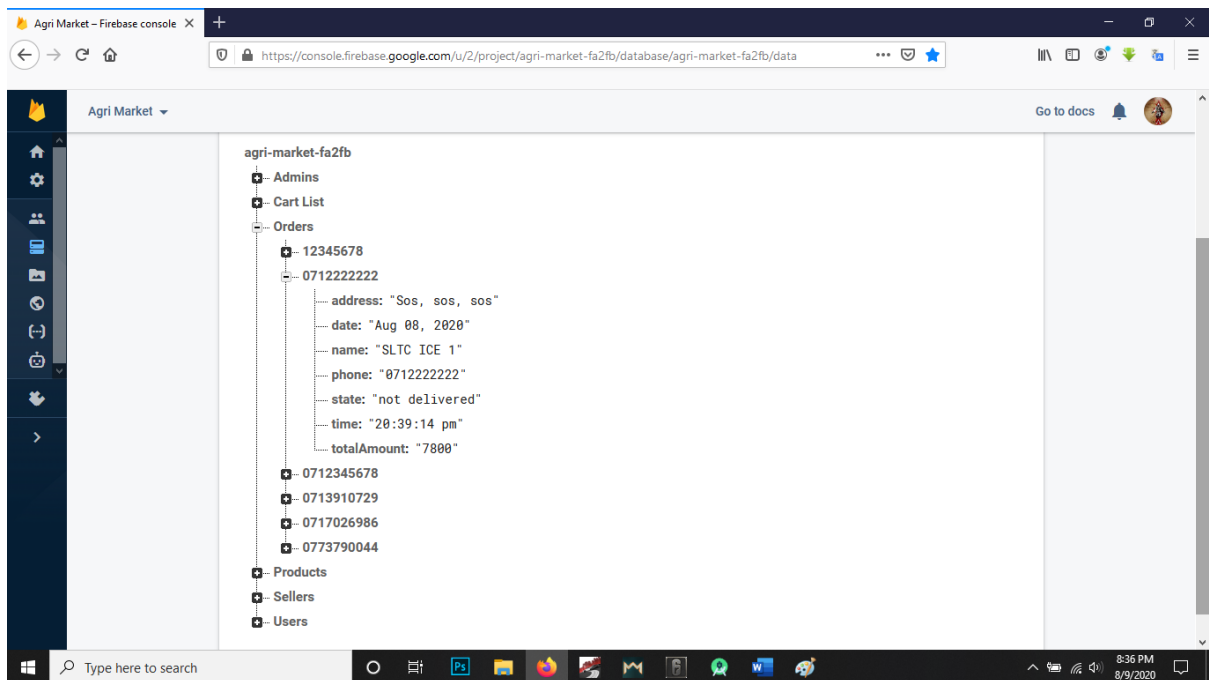
Cart list table (Admin view)



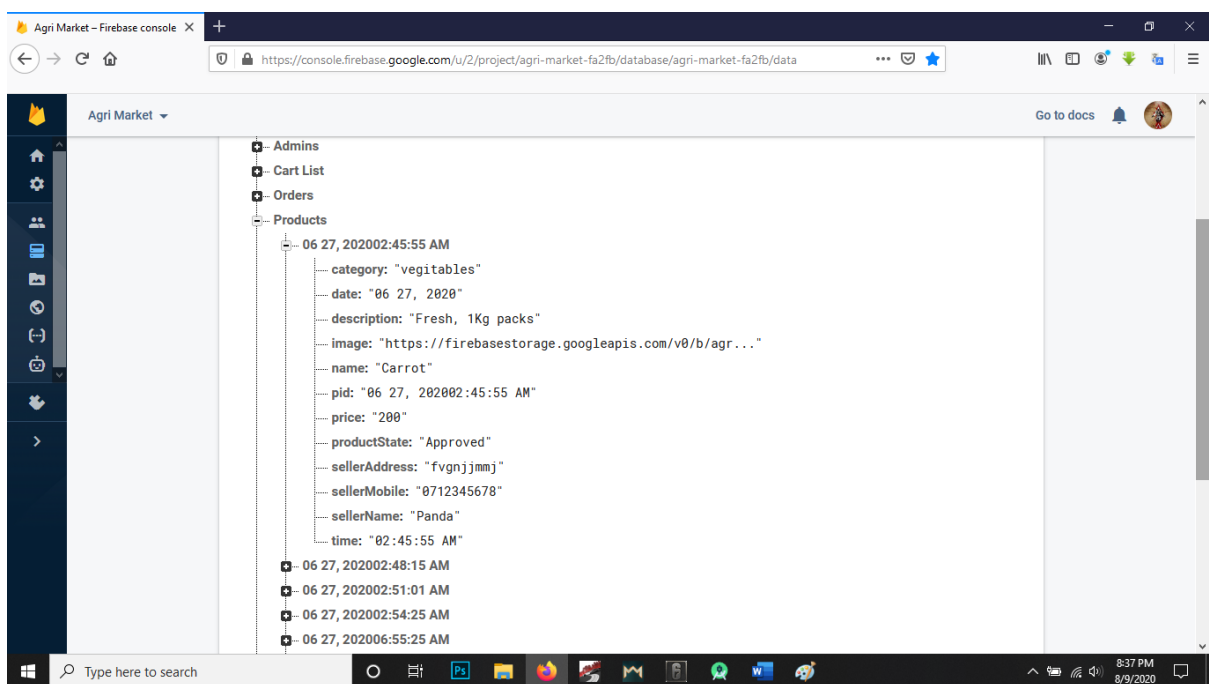
Cart list table (User view)



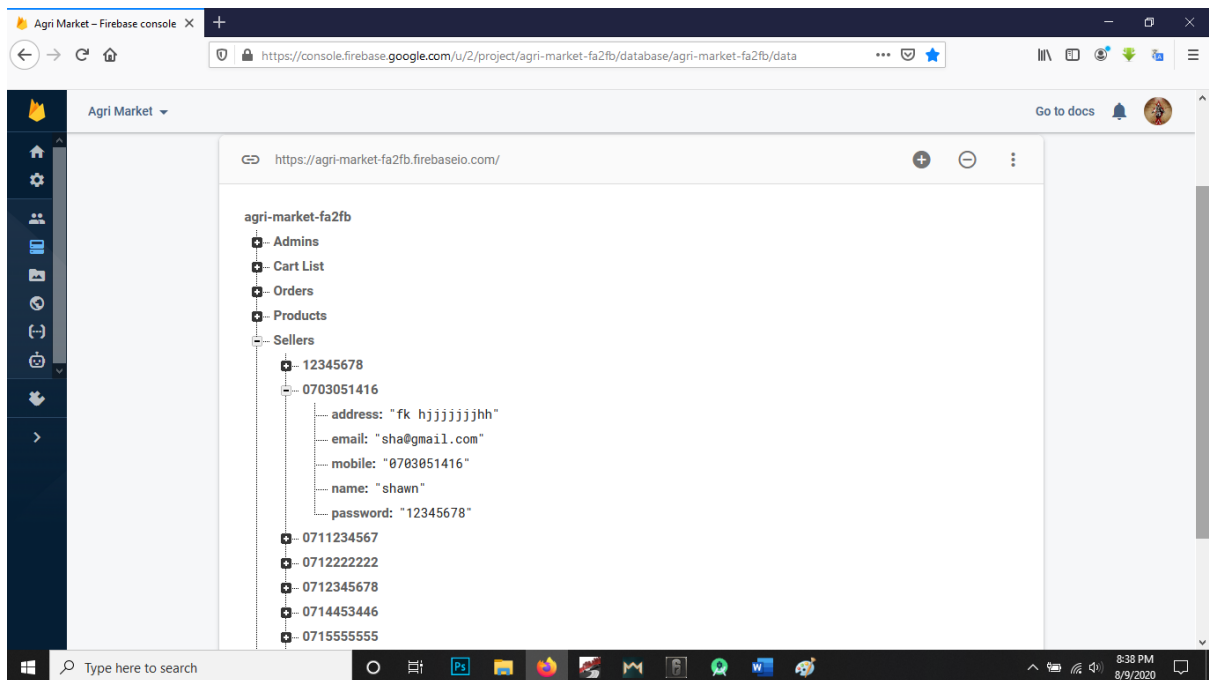
Orders table



Products table



Sellers table



Users table

