LAPORAN PROJECT

“LYT (LockYourTicket)”

Revisi xxx

Disusun Oleh :

Alexander Toar	(00000051653)
Aljevan Komala	(00000044016)
Henry Tantyo Bintang Timur	(00000052755)
Irfan Fari Ramadhan	(00000052592)
Jerrell Susilo	(00000045370)
Thomas Januardy	(00000046001)
Vannes Lie	(00000045860)

Program: Information Systems

Matakuliah: Mobile Application Development

UNIVERSITAS MULTIMEDIA NUSANTARA

TANGERANG

GANJIL 2021/2022

HALAMAN PENGESAHAN PROPOSAL

Disusun oleh tim *Mobile Application Development*

NIM	Nama	Program	No HP
00000051653	Alexander Toar	Sistem Informasi	087880639078
00000044016	Aljevan Komala	Sistem Informasi	087877167318
00000052755	Henry Tantyo Bintang Timur	Sistem Informasi	082138132564
00000052592	Irfan Fari Ramadhan	Sistem Informasi	081214896373
00000046001	Thomas Januardy	Sistem Informasi	08984102704
00000045860	Vannes Lie	Sistem Informasi	08114100350

Topik	**:**	
Judul Proposal (Indonesia)	**:**	
Judul Proposal (Inggris)	**:**	
Usulan Pembimbing	**:**	1. <<L00079 – Agus Sulaiman>>

HALAMAN PENILAIAN REVIEWER

Topik Final (Diisi oleh Dosen) :

Hasil Review Proposal Skripsi (Diisi oleh Dosen)

Tanggal Review	:	
Keputusan	:	Diterima / Revisi
Paraf Reviewer	:	

BAB I
PENDAHULUAN

1. LATAR BELAKANG

Di zaman serba digital yang semakin berkembang, dimana semua hal dapat dilakukan dengan hanya satu tombol pada *smartphone*, sudah bukan lagi zamannya kita datang ke sebuah acara dan membeli tiket di tempat (*On the spot)*. Dulunya ketika kita ingin datang ke sebuah acara, kita harus membeli tiket di *booth* yang disediakan oleh penyelenggara. Metode tersebut memiliki resiko hilangnya tiket akibat keteledoran pembeli. Hal tersebut mengakibatkan pembeli tiket tidak dapat mengikuti acara. Membuat sebuah aplikasi penyediaan tiket *event* secara *online* dapat menjadi solusi untuk mengatasi masalah tiket fisik. Dengan adanya sebuah *platform* menjual tiket *online* ini, pengguna juga tidak perlu untuk mengantri sehingga semuanya menjadi lebih cepat dan efisien.

Seperti yang kita ketahui, masa pandemi mengharuskan kita untuk selalu melakukan protokol covid-19 seperti kebijakan *work from home* dan belajar daring. Di masa pandemi tersebut juga sudah banyaknya perubahan kegiatan, seperti contoh dimana dulunya seminar dilakukan secara *offline*, sekarang seminar sudah beradaptasi menjadi webinar. Dengan demikian hal-hal tersebut menjadi pertanda bahwa segala hal sudah mulai beralih ke daring. Hal ini menjadi pendorong kami dalam membuat aplikasi *online* serupa dimana pengguna dapat melakukan sebuah reservasi atau pembelian tiket tanpa perlu ke tempatnya.

Dengan semua latar belakang yang kami buat, kami berkomitmen untuk membuat sebuah *platform* yang dapat membantu tidak hanya masyarakat awan, tetapi juga *event organizers* agar semua dapat berjalan dengan lancar. Aplikasi ini kami beri nama “LYT (LockYourTicket)”

Melalui aplikasi kami, kami harap dapat memberikan kepuasaan, kemudahan, serta kenyamanan bagi setiap *user* yang menggunakan aplikasi ini dari mencari *event* yang diinginkan sampai membeli tiket *event* tersebut.

2. RUMUSAN MASALAH

Berdasarkan latar belakang yang telah dipaparkan, terdapat beberapa permasalahan yang ditemukan dalam perkembangan sistem aplikasi yang masih ramai digunakan agar

solusi dapat segera diaplikasikan dan dirumuskan, tanpa permasalahan yang berlebih saat sudah ditujukan kepada *end-users*. Permasalahan tersebut antara lain:

- Sistem "antrian" yang memakan banyak waktu.
- Resiko "kehilangan" validitas tiket yang mungkin terjadi (sudah terjual, kerusakan tiket fisik, dll).
- Banyaknya sampah kertas yang bersifat tidak ramah lingkungan.
- Kurangnya keterangan informasi mengenai suatu acara di tiket yang tertera.
- Kurang adanya *platform* (berupa aplikasi berjaringan) yang mewadahi acara-acara sejenis dan menjadikan tempat sebagai daftar informasi ketersediaan suatu acara yang tersedia.

3. TUJUAN DAN MANFAAT

Tujuan :

1. Memberikan informasi kepada masyarakat mengenai *event-event* yang berjalan.
2. Memberikan opsi kepada masyarakat dalam memilih acara.
3. Mempermudah *event organizer* dalam mendistribusikan tiket/reservasi.
4. Membantu *event organizer* untuk melakukan pendataan peserta yang lebih terstruktur.

Manfaat :

1. Supaya masyarakat mendapatkan informasi relevan mengenai *event-event* yang sedang berjalan.
2. Membantu masyarakat untuk
3. Menambah tingkat penjualan dengan cara yang lebih praktis
4. Meminimalisir *human error* dalam pendataan (duplikasi)

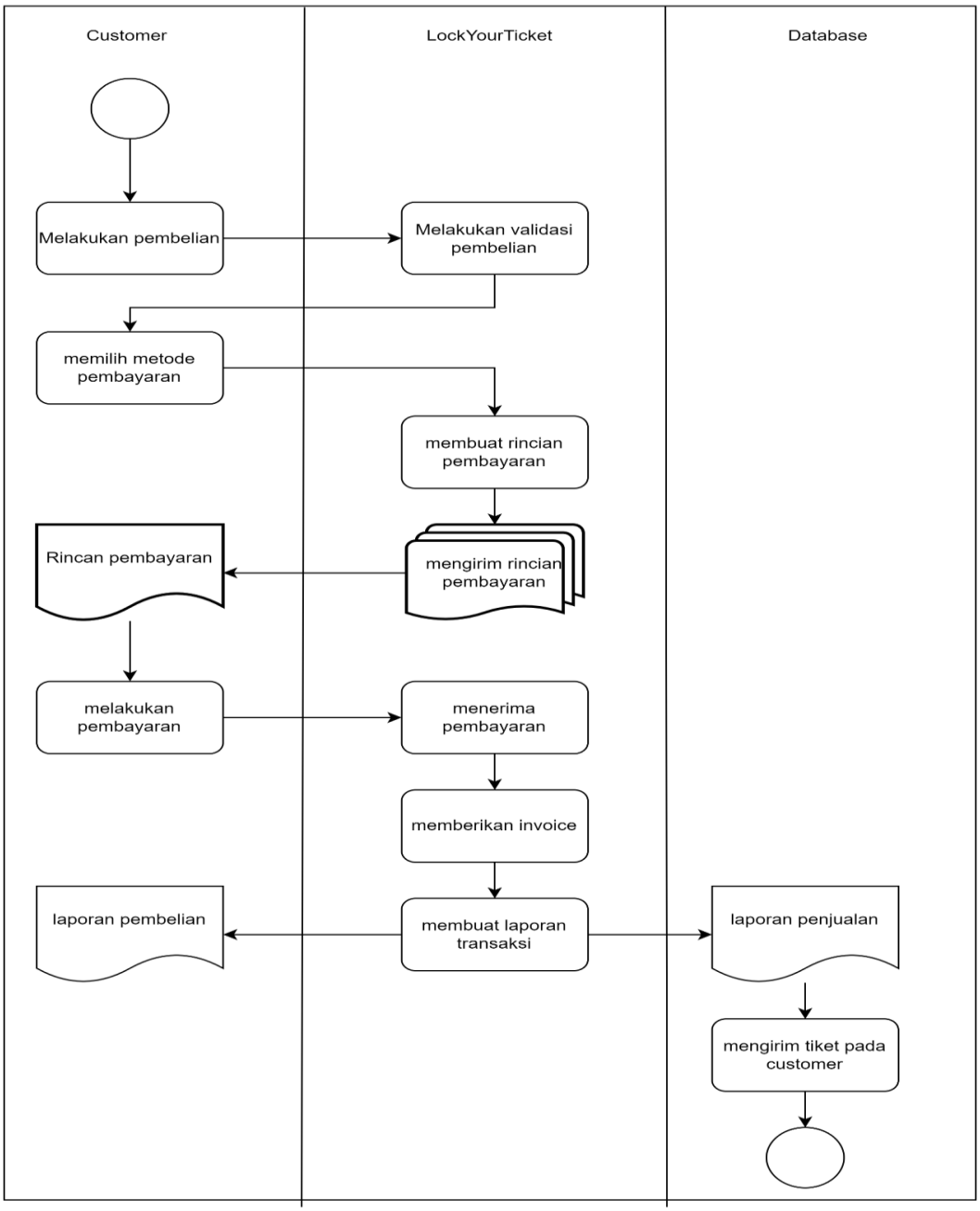
4. PROFIL PERUSAHAAN

LockYourTicket merupakan *startup* aplikasi dari Indonesia yang didirikan oleh sekelompok mahasiswa UMN. Aplikasi tersebut menyediakan perdagangan elektronik berbasis *ticketing* yang menyediakan layanan pemesanan suatu *event* mahasiswa.

- Bagian ini berisi mengenai profil dari perusahaan seperti sejarah perusahaan. Kegiatan bisnis utama perusahaan, lokasi perusahaan, dan informasi relevan lain mengenai perusahaan yang berhubungan dengan project mahasiswa.
- Bagian ini juga menjelaskan objek penelitian.

BAB II
ISI

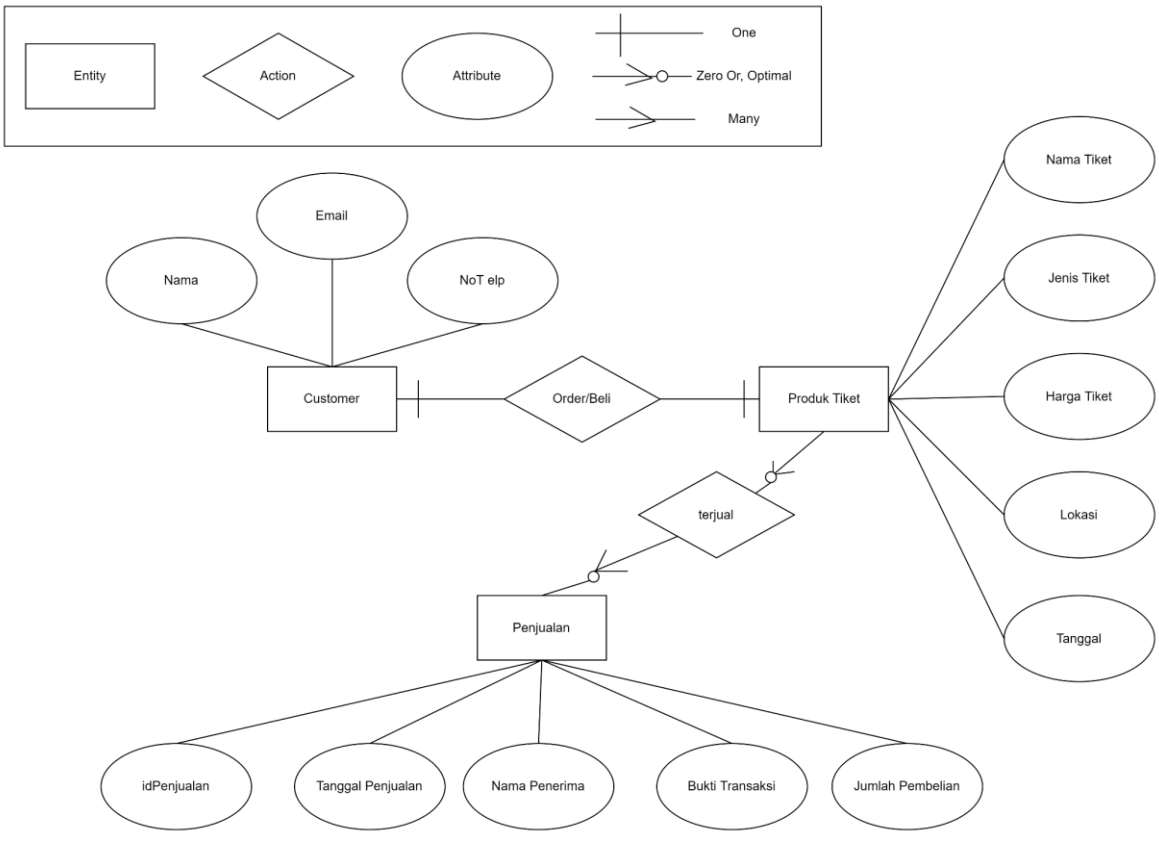
2.1 Proses Bisnis



Proses bisnis adalah suatu gambaran proses berjalannya suatu bisnis dimana terdapat kumpulan atribut atau proses yang memiliki hubungan kuat satu dengan lainnya. Masing-masing proses dibentuk untuk mencapai tujuan yang sama dalam memecahkan suatu masalah. Proses bisnis yang baik adalah proses yang terdiri dari aktivitas yang berurut sesuai dengan runtutan waktu dan terstruktur dengan jelas. Gambar di atas

merupakan proses bisnis yang diterapkan pada aplikasi LockYourTicket. Pertama, pengguna harus mengunduh aplikasi LockYourTicket terlebih dahulu untuk dapat membuka aplikasi tersebut. pada user yang telah memiliki akun dapat langsung melakukan *sign in*. Setelah pengguna masuk ke dalam aplikasi dan melakukan pembelian tiket, dari pihak LockYourTicket akan melakukan validasi pembelian terkait seberapa banyak yang dibeli, jenis tiket apa, dan lain sebagainya. Setelah itu dari pihak *customer* akan memilih metode pembayaran yang akan digunakannya. Kemudian di pihak LockYourTicket akan membuat serta mengirimkan rincian pembayaran kepada *user*. Setelah *user* melakukan pembayaran, pihak LockYourTicket akan memberikan *invoice* dan membuat laporan pembelian yang akan diberikan kepada *customer*. Pihak LockYourTicket membuat laporan penjualan pula untuk dimasukkan ke dalam *database* serta mengirim tiket pada *customer*.

2.2 ER Diagram



ER diagram atau *entity relation diagram* adalah sebuah model yang berisikan data-data yang membentuk suatu model yang memiliki hubungan satu dengan lainnya. Dalam membentuk suatu sistem *database* yang terstruktur dan teratur, diperlukannya

ERD. ERD itu sendiri terdiri dari tiga komponen, yaitu entitas atau kumpulan objek yang ingin diidentifikasi, kemudian ada atribut sebagai penjelas dari dari setiap elemen, dan yang terakhir, yaitu relasi yang merupakan tingkat hubungan atau ketertarikan entitas satu dengan yang lainnya. Untuk menggambarkan atau membuat model atau struktur suatu *database* dengan diagram yang sederhana sehingga memudahkan dalam membuat sebuah *database,* maka buatlah ER diagram diatas. ER diagram kami terdiri dari 3 entitas, yaitu *Customer,* Produk Tiket, dan penjualan.

Dapat kita lihat dari ER Diagram diatas bahwa dimulai dari entitas *customer* yang memiliki atribut nama, alamat *email,* dan nomor telepon. Atribut ini diperlukan untuk melakukan *order* tiket yang nanti akan dijadikan sebagai data *customer*. Kemudian dilanjutkan dengan entitas produk tiket dimana di dalamnya terdapat beberapa atribut , yaitu nama tiket, jenis tiket, harga tiket, lokasi dan tanggal. Nama tiket adalah tiket yang dipilih dan dipesan oleh *customer*. Jenis tiket diartikan bahwa *customer* memesan tiket VVIP, VIP, Tribun, dan lain sebagainya. Dilanjutkan dengan harga tiket, lokasi event akan diselenggarakan, dan tanggal penyelenggaraanya. Setelah produk tiket, kita masuk pada penjualan dimana dalan entitas penjualan tersebut terdapat atribut ‘IdPenjualan’, tanggal penjualan, nama penerima, bukti transaksi, dan jumlah pembelian. ‘IdPenjualan’ merupakan kode unik yang nantinya akan diberikan pada masing-masing tiket. Kemudian tanggal penjualan, hari dimana *customer* membeli tiket tersebut, nama penerima, bukti transaksi dan jumlah total tiket yang dibeli customer.

1. Tabel Customer
   Atribut: Nama
   Atribut: Email
   Atribut: NoTelp

2. Tabel Produk Tiket
   Atribut: Nama Tiket
   Atribut: Jenis Tiket
   Atribut: Harga Tiket
   Atribut: Lokasi
   Atribut: Tanggal

3. Tabel Penjualan

Atribut: idPenjualan

Atribut: Tanggal Penjualan

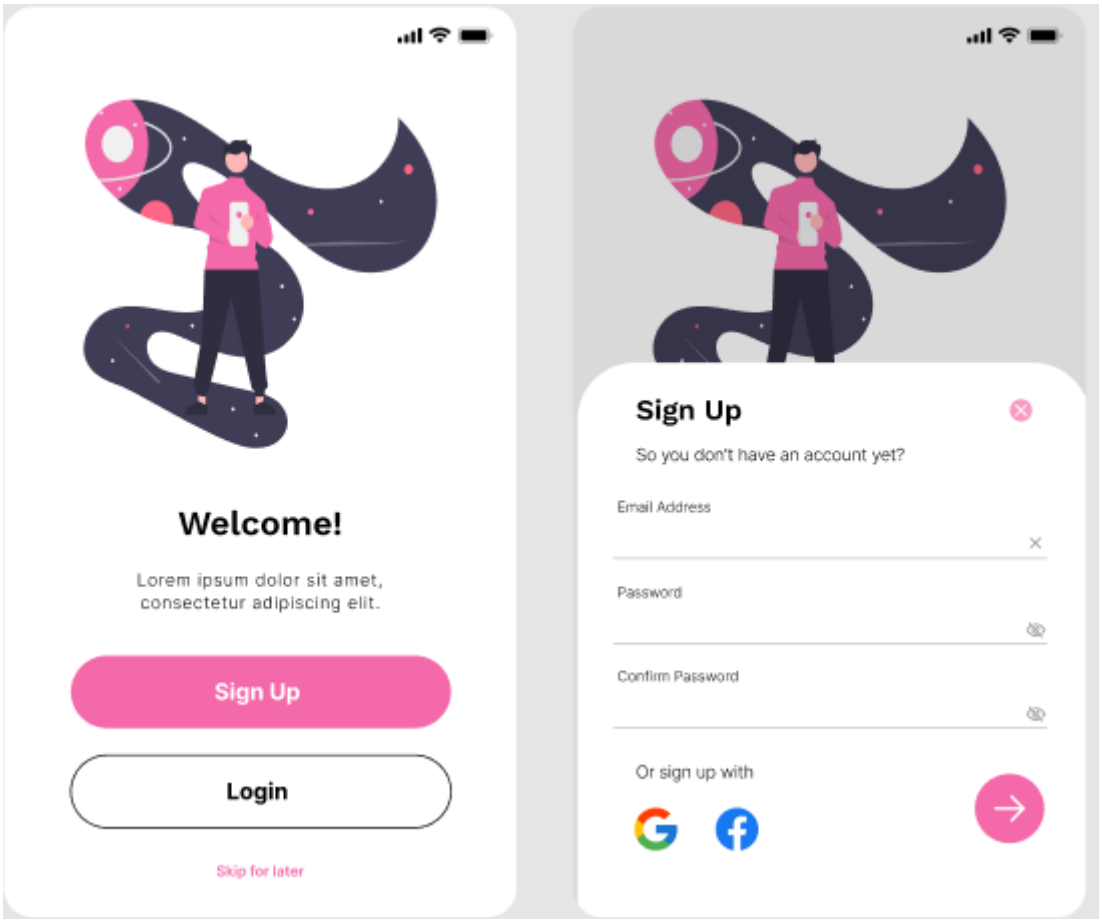
Atribut: Nama Penerima

Atribut: Bukti Transaksi

Atribut: Jumlah Pembelian

2.3 *Layout* Aplikasi

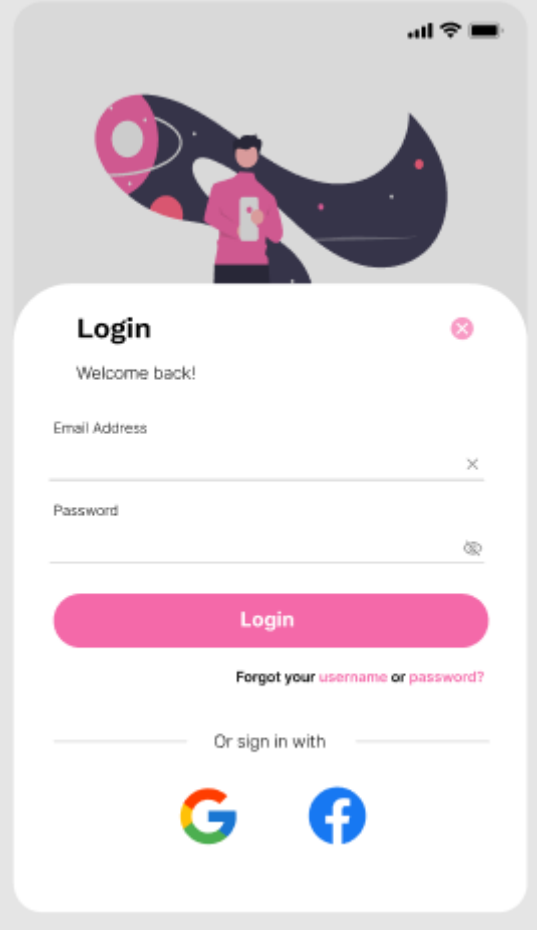
2.3.1 *Page Sign Up* dan *Welcome Page*



Di atas merupakan tampilan *user interface* dari halaman *sign up* dan *welcome page*. Pada tampilan *welcome page*, terdapat 2 *button*, yaitu "Sign Up" dan "Login". *Button* "Sign Up" berfungsi untuk melakukan pendaftaran pertama kali untuk menjadi pengguna atau *user* dari aplikasi ini. Sedangkan untuk *button* "Login" digunakan untuk pengguna yang sudah memiliki akun terdaftar di aplikasi untuk masuk ke aplikasi. Pada tampilan *page sign up*, calon pengguna yang ingin mendaftarkan akun diminta untuk meng-*input* beberapa informasi terkait data diri, seperti alamat email dan password. Calon

pengguna juga dapat mendaftar menggunakan akun Gmail ataupun Facebook dengan mengklik salah satu *button image* Gmail atau Facebook. Setelah selesai memasukkan data diri, pengguna akan dibawa menuju menu *login* untuk masuk ke aplikasi. Apabila data diri yang dimasukkan salah atau kurang lengkap, akan tampil peringatan tentang data diri yang salah.

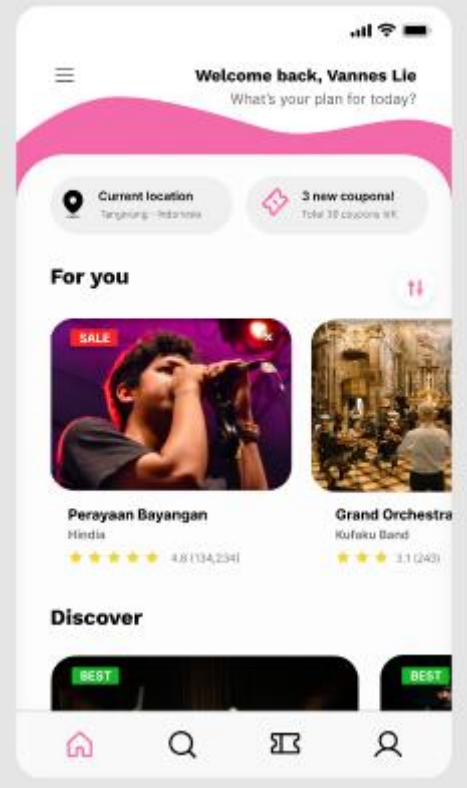
2.3.2 *Page Login*



Di atas merupakan tampilan *user interface* dari halaman *login*. Halaman *login* ini berfungsi bagi pengguna yang sudah memiliki akun terdaftar di aplikasi ini. Pada halaman ini pengguna diminta untuk memasukkan *email* dan *password*, atau juga bisa menggunakan akun Gmail atau Facebook yang sudah terdaftarkan pada *database* aplikasi. Apabila akun yang dimasukkan atau dipilih sesuai dengan yang ada pada database aplikasi, pengguna akan diteruskan menuju halaman *home*. Selain itu, pada halaman ini juga terdapat fitur yang dapat membantu pengguna, apabila pengguna lupa dengan *username* ataupun *password* miliknya. Dengan mengklik fitur

tersebut pengguna akan dibawa menuju halaman tersendiri untuk menyelesaikan masalahnya.

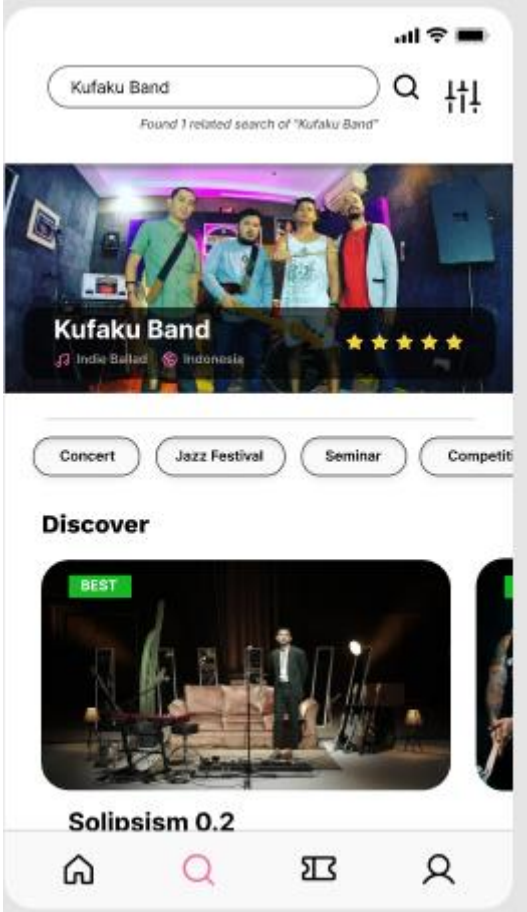
2.3.3 *Home Menu*



Tampilan di atas merupakan tampilan *home menu* yang muncul ketika *user* telah melakukan *login* atau masuk. Pada menu *home* terdapat tampilan dimana nama *user* yang telah login terdapat di paling atas, setelah itu pengguna juga disuguhkan beberapa rekomendasi penjualan tiket yang sedang berlangsung dengan fitur *scroll view*, dimana pengguna dapat menggeser ke kanan dan kiri mengenai penjualan tiket yang sedang berlangsung. Selain itu, pada menu *home* pengguna juga dapat mengatur

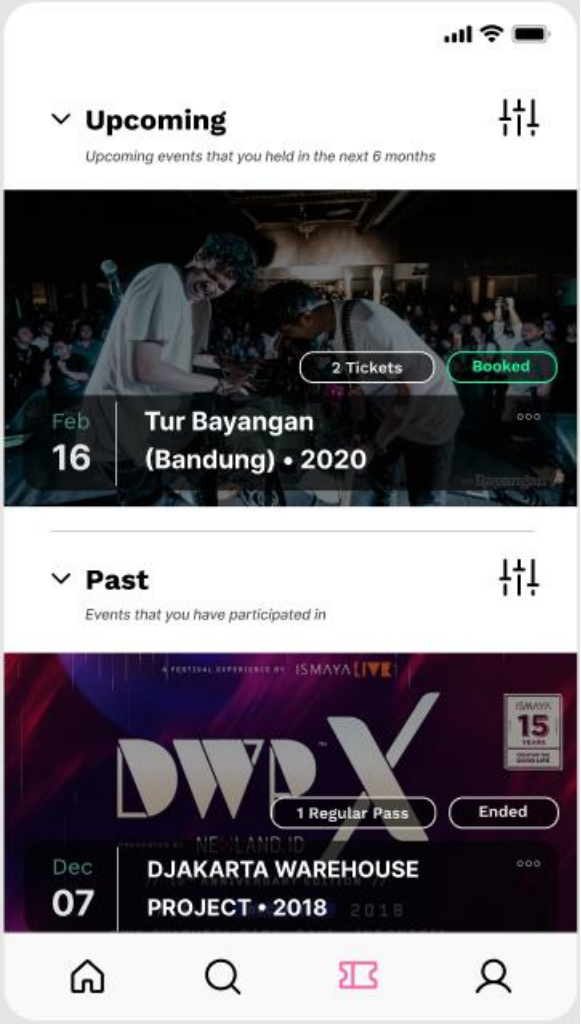
lokasi utama pengguna pada *button* “Current Location”, melihat kupon pada *button* “Coupons” dengan teks “3 new coupons!”.

2.3.4 *Search Ticket Page*



Di atas merupakan tampilan dari menu *search ticket*. Pada menu ini, pengguna dapat mencari tiket dari *event* yang sedang pengguna cari. Bagi pengguna yang sudah mengetahui *event* apa yang sedang mereka cari, pengguna dapat langsung mencari tiket dari *event* tersebut dengan meng-*input* nama *event* pada kolom search. Pada menu ini juga terdapat beberapa fitur lainnya, seperti *filler* dan pilihan beberapa kategori *event* yang ada. Kedua fitur ini dapat digunakan apabila pengguna masih belum mengetahui *event* apa yang ingin dinikmati dan mencari referensi pilihan *event* yang mereka minati untuk ikuti.

2.3.5 *My Events Page*

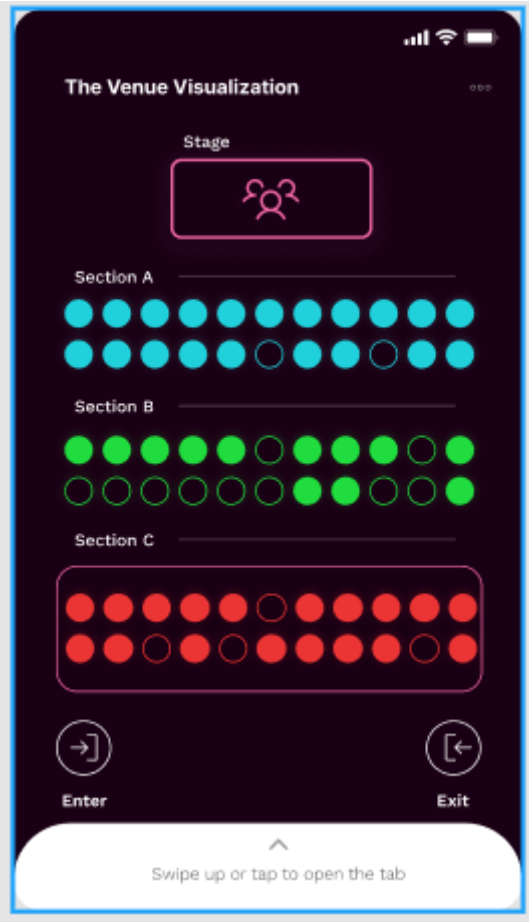


Di atas adalah tampilan dari *menu 'my events'* yang memiliki fungsi dimana pengguna memiliki akses melihat *upcoming event* berdasarkan tiket acara yang telah ia beli. Pengguna juga memiliki akses untuk melihat seluruh transaksi pembelian tiket yang sudah pernah dibeli sebelumnya, dalam kata lain adalah *history* atau riwayat pembelian.

2.3.6 *Event Poster Page*

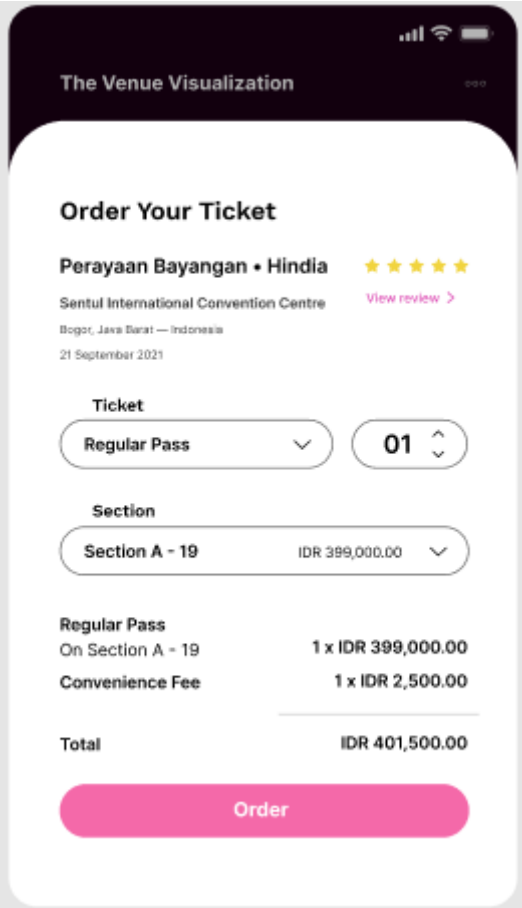
Berikut merupakan tampilan dari poster *event* yang telah dipilih dari *menu search ticket*. Pada bagian ini, pengguna ditampilkan dengan beberapa informasi dan keterangan dari event yang pengguna pilih. Pada halaman ini, terdapat *image button* yang dapat membawa pengguna menuju ke akun Instagram resmi dari *event* yang dipilih. Adanya fitur ini diharapkan agar pengguna dapat mencari informasi secara lebih lengkap langsung dari akun Instagram resmi *event*. Apabila pengguna sudah yakin untuk membeli tiket dari *event* tersebut, pengguna dapat mengklik *button* "Proceed" untuk melanjutkan ke proses pemilihan kursi dan pembayaran tiket.

2.3.7 *Venue Visualization*



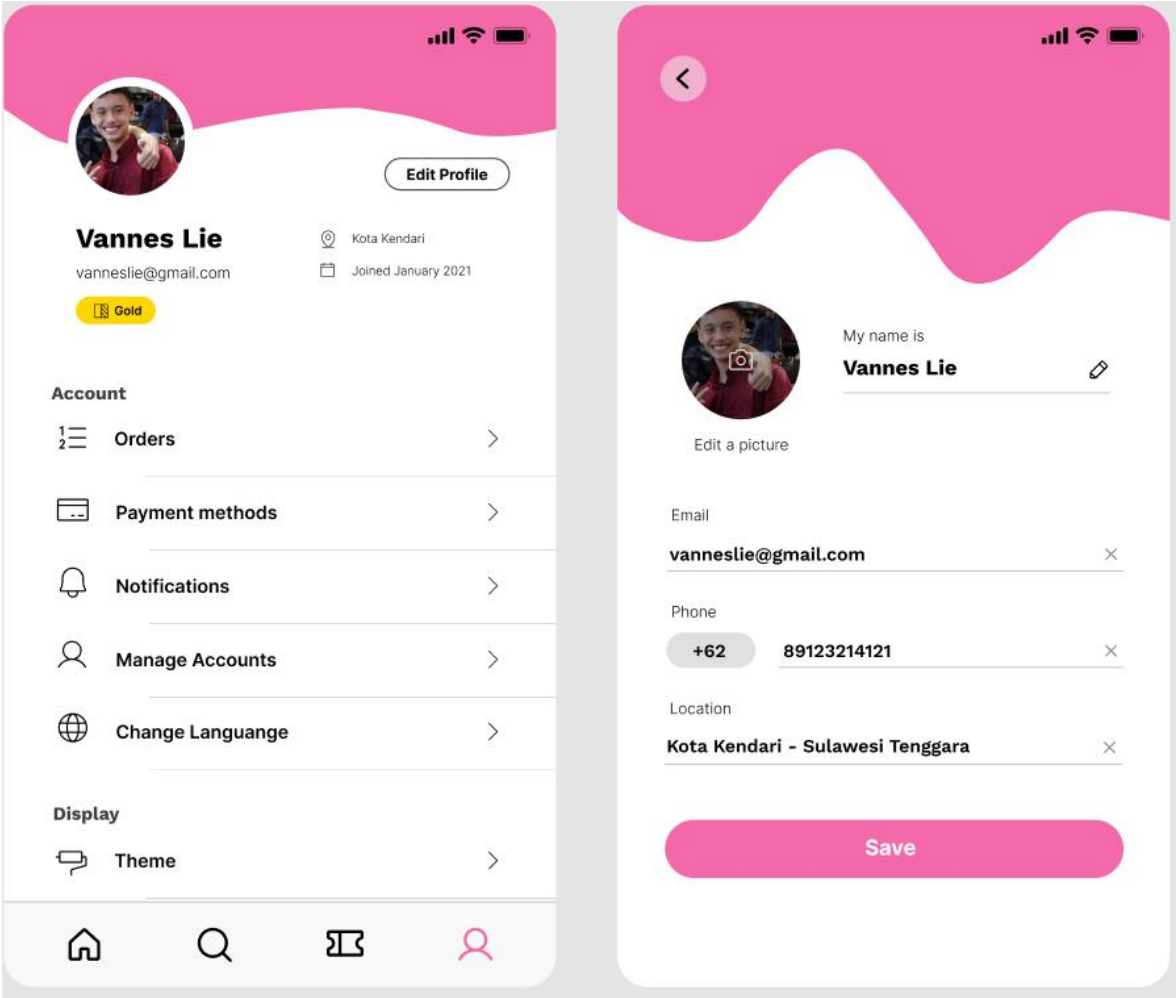
Setelah pengguna memutuskan untuk melanjutkan proses pembelian tiket dari halaman *event poster*, pengguna akan ditampilkan halaman untuk memilih posisi tempat duduk dari *event* yang dipilih. Pada halaman ini, pengguna diberikan visualisasi mengenai denah tempat duduk dari event sehingga pengguna dapat memilih kursi sesuai dengan posisi yang diinginkan. Pengguna dapat memilih posisi kursi dengan mengklik *button* berbentuk bulat. Untuk dapat mengetahui apakah kursi tersebut sudah di-*booking* atau belum, pengguna dapat mengetahuinya dengan melihat apakah bulatan sudah terisi dengan warna atau belum. Apabila bulatan posisi kursi hanya berbentuk lingkaran tanpa ada '*fill color',* maka posisi tersebut masih belum ada yang booking. Setelah selesai memilih posisi kursi, pengguna dapat *swipe up* atau *tap* bagian bawah layar untuk melanjutkan ke tab proses pembelian tiket.

2.3.8 '*Proceed the Ticket' Page*



Pada halaman ini, pengguna akan ditampilkan dengan keterangan total harga dari tiket event yang ingin dibeli. Pengguna juga dapat memilih jenis tiket serta jumlah tiket yang ingin dibeli. Keterangan harga yang ditampilkan merupakan akumulasi dari harga jenis tiket, posisi kursi, jumlah tiket, dan *convenience fee*. Apabila pengguna sudah yakin untuk membeli tiket, pengguna dapat mengklik *button* "Order" untuk melakukan proses pembayaran melalui aplikasi ini.

2.3.9 *Profile Menu*



Dua tampilan di atas merupakan tampilan utama dari *profile menu* dan tampilan ketika *user* mengatur *profile*-nya (*Edit Profile*). Pada menu tersebut, pengguna dapat mengisi data diri mereka pada "Edit Profile". Pada *edit profile,* pengguna akan dapat mengatur identitas diri mereka, mulai dari *profile picture*, nama lengkap, *email*, *phone*, dan *location*. Bagian utama *profile menu* juga memuat sejumlah *submenu* lainnya seperti *Order*, *Payment Methods*, *Notifications*, *Manage Account*, *Change Language*, serta pengguna dapat mengatur *display theme* daripada aplikasi ini.

2.4 *Coding* Aplikasi

Android Studio

2.4.1 *Coding* aplikasi

Sign Up

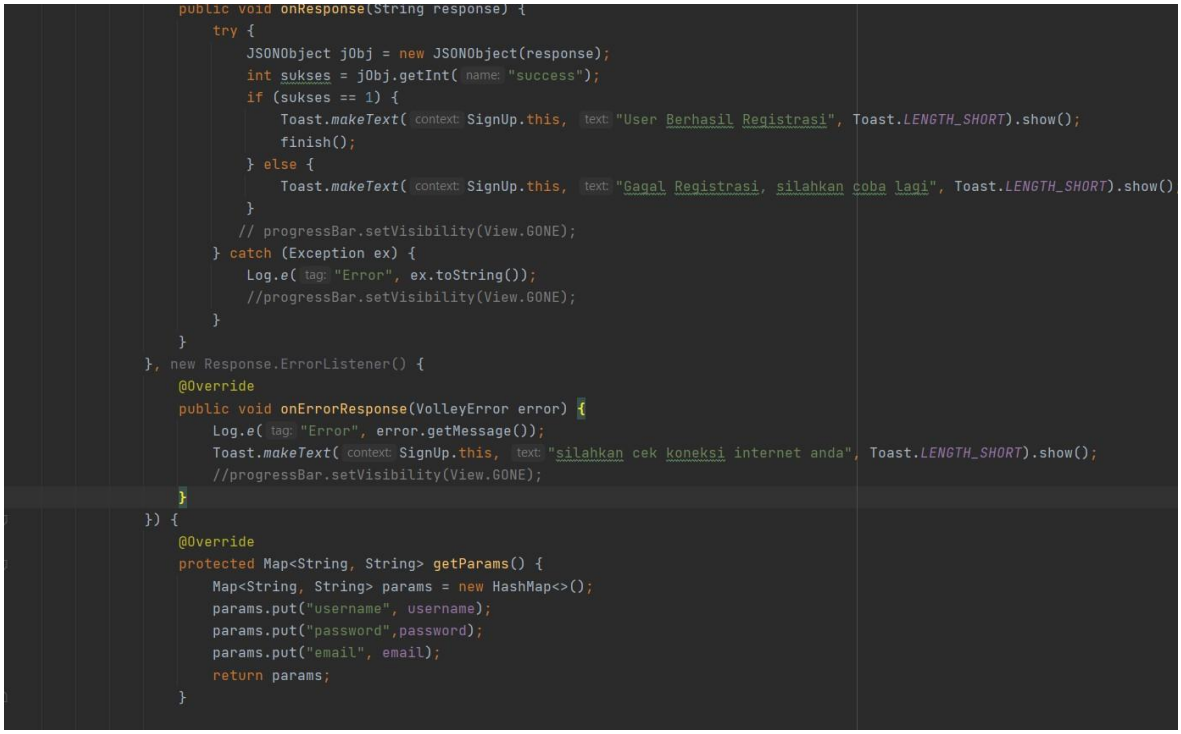
```
public class SignUp extends AppCompatActivity {
    private EditText  etEmail, etPassword, etReenterPassword;

    private ImageButton btnRegister;
    private String URL = "http://192.168.1.5/register2.php";
    private String name, email, password, reenterPassword;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);
        getSupportActionBar().hide();

        etEmail = findViewById(R.id.txtEmail);
        etPassword = findViewById(R.id.txtPassword);
        etReenterPassword = findViewById(R.id.txtConfirmPass);

        btnRegister = findViewById(R.id.next_btn);
        email = password = reenterPassword = "";
    }

    public void save(View view) {

        email = etEmail.getText().toString().trim();
        password = etPassword.getText().toString().trim();
        reenterPassword = etReenterPassword.getText().toString().trim();
        if(!password.equals(reenterPassword)){
            Toast.makeText( context: this,  text: "Password Mismatch", Toast.LENGTH_SHORT).show();
        }
        else if(!email.equals("") && !password.equals("")){
            StringRequest stringRequest = new StringRequest(Request.Method.POST, URL, new Response.Listener<String>() {
                @Override
```

```
                public void onResponse(String response) {
                    if (response.equals("success")) {
                        Toast.makeText( context: SignUp.this,  text: "Register Berhasil", Toast.LENGTH_SHORT).show();
                        Intent i = new Intent( packageContext: SignUp.this, LoginActivity.class);
                        startActivity(i);
                        btnRegister.setClickable(false);
                    } else if (response.equals("failure")) {
                        Toast.makeText( context: SignUp.this,  text: "Register gagal, silahkan coba lagi", Toast.LENGTH_SHORT).show();
                    }
                }, new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
                        Toast.makeText(getApplicationContext(), error.toString().trim(), Toast.LENGTH_SHORT).show();
                    }
                }){
                    @Override
                    protected Map<String, String> getParams() throws AuthFailureError {
                        Map<String, String> data = new HashMap<>();

                        data.put("email", email);
                        data.put("password", password);
                        return data;
                    }
                };
                RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
                requestQueue.add(stringRequest);
            }
        }


}
```

Di atas merupakan *code* kami pada Android Studio untuk menjalankan proses *sign up* atau daftar pada halaman masuk. Pada Java tersebut, kami menggunakan *database* untuk menampung data *sign up* dari *user*, serta kami melakukan *hosting file* PHP kami tersebut pada *localhost*. Pada *code*, kami menggunakan *library volley* untuk *input* data ketika *user* melakukan *sign up*. Dapat dilihat kami menggunakan logika 'IF' dan 'Else' untuk proses *sign up*, dimana ketika *user* berhasil *sign up* akan muncul *toast* "User Berhasil Registrasi", dan apabila *user* tidak mengisi data dengan benar akan *toast* "Gagal Registrasi, silahkan coba lagi". Setelah *user* mengisi data pada *sign up*, *user* akan diarahkan ke halaman *login* dan data di-*input* dengan 'AddUser.php' yang sudah dibuat dalam *hosting*.

Login Activity

```
package com.example.mobileappdev_project;

import ...

public class SignUp extends AppCompatActivity {
EditText edtUser,edtPassword,edtEmail;
String url_add_user ="http://lockyourticket.000webhostapp.com/addUser.php";
String email,username,password;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_up);
        getSupportActionBar().hide();

        ImageButton btn = (ImageButton)findViewById(R.id.next_btn);

        edtUser = (EditText) findViewById(R.id.txtUser);
        edtEmail = (EditText) findViewById(R.id.txtEmail);
        edtPassword = (EditText) findViewById(R.id.txtPassword);

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent( packageContext: SignUp.this, WelcomeActivity.class));
                email = edtEmail.getText().toString();
                username = edtUser.getText().toString();
                password = edtPassword.getText().toString();
                RequestQueue queue = Volley.newRequestQueue( context: SignUp.this);

                StringRequest stringRequest = new StringRequest(Request.Method.POST, url_add_user, new Response.Listener<String>() {
                    @Override
```



```
            public void onResponse(String response) {
                try {
                    JSONObject jObj = new JSONObject(response);
                    int sukses = jObj.getInt( name: "success");
                    if (sukses == 1) {
                        Toast.makeText( context: SignUp.this,  text: "User Berhasil Registrasi", Toast.LENGTH_SHORT).show();
                        finish();
                    } else {
                        Toast.makeText( context: SignUp.this,  text: "Gagal Registrasi, silahkan coba lagi", Toast.LENGTH_SHORT).show();
                    }
                   // progressBar.setVisibility(View.GONE);
                } catch (Exception ex) {
                    Log.e( tag: "Error", ex.toString());
                    //progressBar.setVisibility(View.GONE);
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.e( tag: "Error", error.getMessage());
                Toast.makeText( context: SignUp.this,  text: "silahkan cek koneksi internet anda", Toast.LENGTH_SHORT).show();
                //progressBar.setVisibility(View.GONE);
            }
        }) {
            @Override
            protected Map<String, String> getParams() {
                Map<String, String> params = new HashMap<>();
                params.put("username", username);
                params.put("password",password);
                params.put("email", email);
                return params;
            }
```

Di atas merupakan *coding* pada aplikasi kami untuk bagian ‘LoginActivity’. Pada *syntax* ini, nantinya pada aplikasi akan ditampilkan bagian *login* untuk pengguna yang sudah memiliki akun terdaftar pada aplikasi. Ketika pengguna berhasil memasukkan *email* dan *password* yang tepat, akan muncul *toast* bertuliskan “Success”. Akan tetapi apabila pengguna memasukkan *email* dan *password* yang salah, akan tampil *toast* bertuliskan *failure*. Jika pengguna tidak memasukkan *email* atau *password* pada kolom yang disediakan, akan muncul *toast* bertuliskan “Fields can not be empty”.

Custom Maps

```
package com.example.mobileappdev_project;

import ...

public class CustomMaps extends AppCompatActivity implements OnMapReadyCallback {
    Button  mylocation;
    GoogleMap map;
    LatLng home, position;
    Double latitude, longitude, x, y;
    int PLACE_AUTO = 1;

    @Override
    public void onMapReady(GoogleMap googleMap){
        map = googleMap;
        home = new LatLng( v: -6.257385,  v1: 106.618320);
        map.addMarker(new MarkerOptions().position(home).title("Lokasi Anda!")).showInfoWindow();
        map.moveCamera(CameraUpdateFactory.newLatLng(home));
        map.moveCamera(CameraUpdateFactory.newLatLngZoom(home, v: 16));
        map.setTrafficEnabled(true);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_custom_maps);
        getSupportActionBar().hide();

        try{
            SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map);
            mapFragment.getMapAsync( onMapReadyCallback: this);
        }catch (Exception e){
            Toast.makeText(getApplicationContext(),e.toString(), Toast.LENGTH_LONG).show();
        }
```

```
    mylocation = findViewById(R.id.btnMyLocation);
    mylocation.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            try{
                if (ActivityCompat.checkSelfPermission(getApplicationContext(), Manifest.permission.ACCESS_FINE_LOCATION)!= PackageManager.PERMISSION_GRANTED){
                    Toast.makeText(getApplicationContext(), text: "Error: Tidak ada akses ke GPS!",Toast.LENGTH_SHORT).show();
                    return;
                }
                map.setMyLocationEnabled(true);
                LocationManager locationManager= (LocationManager) getSystemService(LOCATION_SERVICE);
                Criteria criteria = new Criteria();
                //Location location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                Location location = locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                latitude = location.getLatitude();
                longitude = location.getLongitude();
                position = new LatLng(latitude,longitude);
                map.addMarker(new MarkerOptions().position(position).title("My Location Now")).showInfoWindow();
                map.animateCamera(CameraUpdateFactory.newLatLng(position));
                map.animateCamera(CameraUpdateFactory.newLatLngZoom(position, v: 18));
            }catch (Exception e){
                Toast.makeText(getApplicationContext(),e.toString(),Toast.LENGTH_LONG).show();
            }
        }
    });
}

}
```

Pada gambar di atas merupakan *code* dari ‘Custom Maps’, pada saat *user* masuk pada aplikasi di lokasi tertentu, secara otomatis lokasi tersebut sesuai dengan pada saat *user* berada. Tombol tersebut berada pada *menu home*, dimana pada *button* tersebut bernama “Current Location”. Pada saat *user* menekan tombol tersebut, muncul tampilan peta dimana *user* tersebut berada.

Main Activity

```
package com.example.mobileappdev_project;

import ...


public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getSupportActionBar().hide();


        Home_Menu frag1 = new Home_Menu();
        frag1.setArguments(getIntent().getExtras());

        FragmentManager fm = getSupportFragmentManager();
        FragmentTransaction ft =fm.beginTransaction();

        ft.add(R.id.frame1,frag1);
        ft.replace(R.id.frame1,frag1);
        ft.commit();


        ImageButton BtnHome = (ImageButton) findViewById(R.id.BtnHome);
        BtnHome.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Home_Menu frag1 = new Home_Menu();
                frag1.setArguments(getIntent().getExtras());

                FragmentManager fm = getSupportFragmentManager();
```

```
            FragmentTransaction ft =fm.beginTransaction();

            ft.add(R.id.frame1,frag1);
            ft.replace(R.id.frame1,frag1);
            ft.commit();
        }
    });
    ImageButton BtnSearch = (ImageButton) findViewById(R.id.BtnSearch);
    BtnSearch.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            SearchMenu frag1 = new SearchMenu();
            frag1.setArguments(getIntent().getExtras());

            FragmentManager fm = getSupportFragmentManager();
            FragmentTransaction ft = fm.beginTransaction();

            ft.add(R.id.frame1, frag1);
            ft.replace(R.id.frame1, frag1);
            ft.commit();


        }
    });

    ImageButton BtnHistory = (ImageButton) findViewById(R.id.BtnHistory);
    BtnHistory.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            HistoryMenu frag1 = new HistoryMenu();
            frag1.setArguments(getIntent().getExtras());

            FragmentManager fm = getSupportFragmentManager();
```

```
        ImageButton BtnHistory = (ImageButton) findViewById(R.id.BtnHistory);
        BtnHistory.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                HistoryMenu frag1 = new HistoryMenu();
                frag1.setArguments(getIntent().getExtras());

                FragmentManager fm = getSupportFragmentManager();
                FragmentTransaction ft = fm.beginTransaction();

                ft.add(R.id.frame1, frag1);
                ft.replace(R.id.frame1, frag1);
                ft.commit();

            }
        });


        ImageButton btnprofile = (ImageButton)findViewById(R.id.BtnProfile);
        btnprofile.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent( packageContext: MainActivity.this, Profile.class));
            }
        });

    }
}
```

Pada ‘Main Activity’, terdapat beberapa dari *home menu* kami. Pada *menu home* tersebut, terdiri dari 3 *fragment* utama, yaitu *home menu*, *search menu*, dan *ticket history*. Dari *code* di atas, tersebut dapat membuat *fragment* dari masing-masing *fragment* utamanya.

Poster Screen

```
public class PosterScreen extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_screen_poster);
        getSupportActionBar().hide();

        ImageButton btn = (ImageButton)findViewById(R.id.btnProses);

        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent( packageContext: PosterScreen.this, EditPembayaran.class));
            }
        });

       ImageButton ig = (ImageButton) findViewById(R.id.btnInstagram);
        ig.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getApplicationContext(), WebViewIG.class));
            }
        });
```

```
        ImageButton btnBookmark = (ImageButton)findViewById(R.id.btnBookmark);

        btnBookmark.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent( packageContext: PosterScreen.this, BookmarkReview.class));
            }
        });

        ImageButton btnback = (ImageButton) findViewById(R.id.btnBack);
        btnback.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startActivity(new Intent( packageContext: PosterScreen.this, MainActivity.class));
            }
        });

    }
}
```

Di atas merupakan *coding* dari tampilan ‘Poster Screen’, yang berfungsi untuk menampilkan informasi terkait dari *event* yang akan dipilih oleh pengguna. Pada bagian ini juga, diberikan *fungsi intent* untuk dapat menampilkan ‘WebView’ ke akun Instagram dari *event* tersebut untuk dapat mencari informasi lebih lanjut.

WebView & ViewPager

```java
public class WebViewIG extends AppCompatActivity {
    ViewPager mViewPager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_web_view_ig);
        getSupportActionBar().hide();

        mViewPager = (ViewPager) findViewById(R.id.pager);
        mViewPager.setAdapter(new ViewPagerAdapter(getSupportFragmentManager()));
    }

    public class ViewPagerAdapter extends FragmentPagerAdapter {
        public ViewPagerAdapter(FragmentManager fm) {
            super(fm, BEHAVIOR_RESUME_ONLY_CURRENT_FRAGMENT);}

        @Override
        public int getCount() { return 3;
        }
        public Fragment getItem(int position) {
            if (position == 0){
                return new FragmentIG();
            } else if (position == 1) {
                return new FragmentPoster();
            }  else {
                return new FragmentYoutube();
            }
        }
```

```java
public class FragmentYoutube extends Fragment {

    WebView webView3;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.activity_fragment_youtube, container,  attachToRoot: false);

        webView3 = (WebView) v.findViewById(R.id.webView3);
        webView3.setWebViewClient(new WebViewClient());
        webView3.getSettings().setJavaScriptEnabled(true);
        webView3.loadUrl("https://www.youtube.com/channel/UC_WMZDQv29eJBMpv9PwWazw");

        return v;
    }
}
```

Di atas merupakan tampilan *coding* untuk bagian ‘WebView’ dan ‘ViewPager’ yang digunakan pada halaman ‘Poster Event’. WebView digunakan untuk menampilkan halaman *web* dari akun Instagram penyelenggara *event* dan akun Youtube *event* tersebut. Saat kita meng-klik tombol instagram yang terdapat pada layar, maka kita akun langsung masuk ke halaman instagram dari event yang kita pilih. Begitu juga dengan halaman

youtube. Kedua aplikasi tersebut dapat diakses dengan menggunakan ‘WebView’. ‘ViewPager’ digunakan untuk menggeser layar ke kanan maupun ke kiri, sehingga dari halaman Instagram, saat kita meng-slide layar ke kiri, kita langsung bisa masuk ke halaman Youtube.

Bookmark (SQLite)

```
private SQLiteDatabase database;

public void open() throws SQLException {
    database = this.getWritableDatabase();
}

private String[] allColumns =
        {COLUMN_ID, COLUMN_JUDULNOTES, COLUMN_KONTENNOTES};

private Notes cursorToNotes(Cursor cursor){
    Notes notes = new Notes();

    notes.setID(cursor.getLong( columnIndex: 0));
    notes.setJudulNotes(cursor.getString( columnIndex: 1));
    notes.setKontenNotes(cursor.getString( columnIndex: 2));
    return notes;
}

public void createNotes(String JudulNotes, String KontenNotes) {
    ContentValues values = new ContentValues();
    values.put(COLUMN_JUDULNOTES, JudulNotes);
    values.put(COLUMN_KONTENNOTES, KontenNotes);

    database.insert(TABLE_NAME,  nullColumnHack: null, values);
}
```

```
public Notes getNotes(Long id){
    Notes notes = new Notes();

    Cursor cursor = database.query(TABLE_NAME,allColumns, selection: "_id="+id, selectionArgs: null, groupBy: null, having: null, orderBy: null);
    cursor.moveToFirst();
    cursor.close();
    return notes;
}

public ArrayList<Notes> getAllNotes(){
    ArrayList<Notes> daftarnotes = new ArrayList<>();
    Cursor cursor = database.query(TABLE_NAME,allColumns, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null);
    cursor.moveToFirst();
    while (!cursor.isAfterLast()){
        Notes notes = cursorToNotes(cursor);
        daftarnotes.add(notes);
        cursor.moveToNext();
    }
    cursor.close();
    return daftarnotes;
}

public void updateNotes(Notes notes){
    String filter = "_id="+ notes.getID();
    ContentValues args = new ContentValues();
    args.put(COLUMN_JUDULNOTES, notes.getJudulNotes());
    args.put(COLUMN_KONTENNOTES, notes.getKontenNotes());

    database.update(TABLE_NAME, args, filter,  whereArgs: null);
}
```

```
    public void deleteNotes(Long id){
        String filter = "_id="+id;

        database.delete(TABLE_NAME, filter,  whereArgs: null);
    }
}
```

Selain fungsi ‘WebView’ dan ‘ViewPager’, terdapat sebuah fitur *bookmark* yang menggunakan sistem dari ‘SQLite’, dengan fungsi ‘DBHandler’. Hal ini memungkinkan pengguna untuk melakukan penyimpanan *event* atau acara dalam bentuk *bookmark* ke dalam penyimpanan lokal maupun akun yang telah terdaftarkan pada aplikasi LockYourTicket. Cara pengoperasiannya pun sangat mudah, pengguna hanya harus membuat sebuah *list* baru dengan judul dan deskripsi apa yang ingin dicantumkan pada *bookmark* yang ingin disimpan. Ketika disimpan, sebuah *event* baru akan muncul pada daftar *bookmark* yang dimiliki oleh suatu akun pengguna tersebut. Pengguna juga dapat melakukan penyuntingan hingga penghapusan suatu *event* di *bookmark* dengan cara menahan salah satu *event* sampai keluar *pop-up* berupa pilihan menu untuk menyunting (“Edit”) dan menghapus (“Hapus”).

Pembayaran

```
public class Pembayaran extends AppCompatActivity {
    EditText edtOrder;
    double jumlahtiket;
    double hargatiket = 399000;
    String jumlaht;
    String url_add_order = "http://192.168.1.7/OrderTiket.php";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pembayaran);
        getSupportActionBar().hide();

        ImageButton order = findViewById(R.id.OrderID);

        edtOrder = findViewById(R.id.edtJumlah);

    }
        public void Order(View view) {
            jumlaht = edtOrder.getText().toString();
            Intent i = getIntent();
            String jenistiket = i.getStringExtra( name: "jenistiket");
            double jt = Double.parseDouble(jumlaht);
            double totalharga = jt*hargatiket;
            if(jumlaht.equals("")){
                Toast.makeText( context: this,  text: "Password Mismatch", Toast.LENGTH_SHORT).show();
            }
            else if(!jumlaht.equals("") ){
                StringRequest stringRequest = new StringRequest(Request.Method.POST, url_add_order, new Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        if (response.equals("success")) {
                            Toast.makeText( context: Pembayaran.this,  text: "Order berhasil, Silahkan Konfirmasi Pembayaran", Toast.LENGTH_SHORT).show();
```

```
                        } else if (response.equals("failure")) {
                            Toast.makeText( context: Pembayaran.this,  text: "Order gagal, silahkan coba lagi", Toast.LENGTH_SHORT).show();             }
                    }
                }, new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
                        Toast.makeText(getApplicationContext(), error.toString().trim(), Toast.LENGTH_SHORT).show();
                    }
                }){
                    @Override
                    protected Map<String, String> getParams() throws AuthFailureError {
                        Map<String, String> data = new HashMap<>();

                        data.put("JenisTiket",jenistiket);
                        data.put("JumlahTiket", String.valueOf(jt));
                        data.put("TotalHarga",String.valueOf(totalharga));
                        return data;
                    }
                };
                RequestQueue requestQueue = Volley.newRequestQueue(getApplicationContext());
                requestQueue.add(stringRequest);
            }
        }
```

Code di atas merupakan bagian dari pembayaran. Pada saat *user* mengklik tombol tiket tersebut, proses selanjutnya akan muncul tampilan poster dari tiket tersebut. Dari poster tersebut, terdapat tombol “Proceed” yang dapat ditekan. Setelah *user* menekan tombol tersebut, akan dilanjutkan pada proses validasi pembayaran. *User* kemudian menekan tombol “Order” untuk melakukan proses pembayaran tersebut.

'ListView' Ticket

```
public class HistoryMenu extends AppCompatActivity {
    ListView lv;
    ArrayList<HashMap<String,String>> list_anggota;
    String url_get_mahasiswa="http://192.168.1.5/getTiket.php";

    private static final String TAG_DATA="data";
    private static final String TAG_ORDER_ID = "OrderID";
    private static final String TAG_JENISTIKET="JenisTiket";
    private static final String TAG_JUMLAHTIKET="JumlahTiket";
    private static final String TAG_TOTAL_HARGA = "TotalHarga";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_history_menu);
        getSupportActionBar().hide();

        list_anggota=new ArrayList<>();
        lv=findViewById(R.id.listView);


        RequestQueue queue = Volley.newRequestQueue( context: HistoryMenu.this);
        StringRequest stringRequest = new StringRequest(Request.Method.POST, url_get_mahasiswa, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {
                    JSONObject jObj = new JSONObject(response);
                    JSONArray member = jObj.getJSONArray(TAG_DATA);
```

```
                    for (int i = 0; i < member.length(); i++) {
                        JSONObject a = member.getJSONObject(i);
                        String orderid = a.getString(TAG_ORDER_ID);
                        String jenistiket = a.getString(TAG_JENISTIKET);
                        String jumlahtiket = a.getString(TAG_JUMLAHTIKET);
                        String totalharga = a.getString(TAG_TOTAL_HARGA);

                        HashMap<String, String> map = new HashMap<>();
                        map.put("OrderID", orderid);
                        map.put("JenisTiket", "Jenis : "+jenistiket);
                        map.put("JumlahTiket", "Jumlah : "+jumlahtiket);
                        map.put("TotalHarga","Rp "+totalharga);

                        list_anggota.add(map);
                        String[] from = {"OrderID","JenisTiket", "JumlahTiket", "TotalHarga"};
                        int[] to = {R.id.edtOrderID,R.id.edtJenis, R.id.edtJumlahTiket, R.id.edtTotalHarga};

                        ListAdapter adapter = new SimpleAdapter(
                                context: HistoryMenu.this, list_anggota, R.layout.list_tiket,
                                from, to);
                        lv.setAdapter(adapter);
                        lv.setOnItemLongClickListener(new AdapterView.OnItemLongClickListener() {
                            @Override
                            public boolean onItemLongClick(AdapterView<?> parent, View view, int position, long id) {
                                String nomor = list_anggota.get(position).get(TAG_ORDER_ID);
                                Intent i = new Intent(getApplicationContext(),DeleteTiket.class);
                                i.putExtra( name: "OrderID", nomor);
                                startActivity(i);

                                return true;
                            }
                        });
                    }
```

```
                        return true;
                    }
                });
            }
        }
        catch (Exception ex) {
            Log.e( tag: "Error", ex.toString());

        }
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Log.e( tag: "Error", error.getMessage());
        Toast.makeText( context: HistoryMenu.this,  text: "silahkan cek koneksi internet anda", Toast.LENGTH_SHORT).show();
        finish();
    }
});
queue.add(stringRequest);

}
}
```

Pada Java ‘ListView’ *Ticket* kami, memuat *coding* berisikan logika pada ‘ListView’ yang sudah terhubung pada *database* kami. Kami menggunakan ‘getMahasiswa.php’ yang sudah kami *hosting*. Pada ListView akan menampilkan beberapa data yang telah dipesan sebelumnya dan sudah dideklarasikan dengan ‘String’ pada Java tersebut, yaitu ‘jenisTiket’, ‘jumlahTiket’, dan ‘hargaTiket’. ‘ListView’ juga dapat menghapus data yang telah di-*input* melalui *order* tiket sebelumnya.

2.4.2 Coding PHP

conn.php

```php
<?php
// Create 4 variables to store these information
$server="localhost";
$username="root";
$password="";
$database = "week10";
// Create connection
$conn = new mysqli($server, $username, $password, $database);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
?>
```

Di atas merupakan hasil *coding* yang berfungsi untuk menghubungkan ke *database* aplikasi LockYourTicket pada 'PHPMyAdmin'.

Login2.php

```php
<?php
// Check if email and password are set
if(isset($_POST['email']) && isset($_POST['password'])){
    // Include the necessary files
    require_once "conn2.php";
    require_once "validate.php";
    // Call validate, pass form data as parameter and store the returned value
    $email = validate($_POST['email']);
    $password = validate($_POST['password']);
    // Create the SQL query string
    $sql = "select * from users where email='$email' and password='" . md5($password) . "'";
    // Execute the query
    $result = $conn->query($sql);
    // If number of rows returned is greater than 0 (that is, if the record is found), we'll print "success", otherwise "failure"
    if($result->num_rows > 0){
        echo "success";
    } else{
        // If no record is found, print "failure"
        echo "failure";
    }
}
?>
```

Coding di atas berfungsi untuk menghubungkan ke *database* aplikasi pada saat *login*. Sehingga pada saat *login* menggunakan akun, dapat diketahui apakah akun yang dimasukkan sesuai dengan akun yang ada pada *database* aplikasi.

Register2.php

```php
<?php
$_POST['email'] = 'alexander@gmail.com';
$_POST['password'] = 'Toartoar';
if(isset($_POST['email']) && isset($_POST['password'])){
    // Include the necessary files
    require_once "conn2.php";
    require_once "validate.php";
    // Call validate, pass form data as parameter and store the returned value
    $email = validate($_POST['email']);
    $password = validate($_POST['password']);
    // Create the SQL query string. We'll use md5() function for data security. It calculates and returns the MD5 hash of a string
    $sql = "insert into users values('','$email', '" . md5($password) . "')";
    // Execute the query. Print "success" on a successful execution, otherwise "failure".
    if(!$conn->query($sql)){
        echo "failure";
    }else{
        echo "success";
    }
}
?>
```

Coding di atas berfungsi untuk memasukkan informasi data akun pengguna ke dalam *database* di MySQL.

OrderTiket.php

```php
<?php
$_POST['jenistiket'] = 'Section A';
$_POST['jumlahtiket'] = '2';
$_POST['totalharga'] = '250000';
if(isset($_POST['jenistiket']) && isset($_POST['jumlahtiket']) && isset($_POST['totalharga'])){
    // Include the necessary files
    require_once "conn2.php";
    require_once "validate.php";
    // Call validate, pass form data as parameter and store the returned value
    $jenistiket = validate($_POST['jenistiket']);
    $jumlahtiket = validate($_POST['jumlahtiket']);
    $totalharga =  validate($_POST['totalharga']);

    $sql = "insert into ordertable values('','$jenistiket', '$jumlahtiket ','$totalharga')";
    // Execute the query. Print "success" on a successful execution, otherwise "failure".
    if(!$conn->query($sql)){
        echo "failure";
    }else{
        echo "success";
    }
}
?>
```

Berikutnya merupakan *file* PHP untuk melakukan proses pembelian tiket dengan memasukkan data yang di-*input* pengguna ke dalam database aplikasi. Adapun data yang dimasukkan adalah jenis tiket, jumlah tiket, dan total harga dari order yang dilakukan oleh pengguna.

DeleteTicket.php

```php
<?php
include "conn2.php";

    if(isset($_POST['OrderID'])){
        $OrderID=$_POST['OrderID'];

        $q=mysqli_query($conn,"DELETE FROM ordertable WHERE OrderID='$OrderID'");
        $response=array();

        if($q){
            $response["success"]=1;
            $response["message"]="Data berhasil di hapus";
            echo json_encode($response);
        }else{
            $response["success"]=0;
            $response["message"]="data gagal di hapus";
            echo json_encode($response);
        }
    }else{
        $response["success"]=-1;
        $response["message"]="Data kosong";
        echo json_encode($response);
    }
?>
```

Pada *file* PHP ini, dapat digunakan untuk melakukan penghapusan tiket yang sudah dipesan oleh pengguna. Proses penghapusan atau *delete* dilakukan sesuai dengan data tiket yang dipesan oleh pengguna. PHP di atas berfungsi untuk menghapus *order* tiket yang telah dilakukan sebelumnya. terdapat tombol “Hapus” dan “Batal” pada layar aplikasi. Pada saat kita klik tombol hapus, maka akan muncul pesan “Data berhasil dihapus” dan apabila kita klik tombol “Batal” akan muncul pesan “data gagal dihapus” dan selain itu, akan muncul pesan “Data kosong”.

BAB III
PENUTUP

3.1 Peran Anggota

Berikut merupakan peran dari masing-masing anggota didalam kelompok LockYourTicket:

1. Alexander Toar - 00000051653

   Bertugas dalam membuat database aplikasi, menyusun laporan, pembuatan *file* PHP

2. Aljevan Komala - 00000044016

   Bertugas menyusun laporan, mendesain UI, membuat ERD, dan Proses Bisnis

3. Henry Tantyo Bintang Timur - 00000052755

   Bertugas menyusun laporan, mendesain UI, membuat ERD, dan Proses Bisnis

4. Irfan Fari Ramadhan - 00000052592

   Bertugas dalam membuat database aplikasi, menyusun laporan, pembuatan file PHP

5. Jerrell Susilo - (00000045370)

   Bertugas menyusun laporan, mendesain UI, membuat ERD, dan proses bisnis

6. Thomas Januardy - 00000046001

   Bertugas dalam mendesain UI dan membuat database aplikasi, merapikan laporan.

7. Vannes Lie - 00000045860

   Bertugas menyusun laporan, mendesain UI, membuat ERD, dan proses bisnis.

3.2 Kesimpulan

LockYourTicket adalah perusahaan *startup* berbasis aplikasi penjualan tiket berbasis *online* yang dapat membantu user dalam membeli tiket tanpa perlu keluar rumah. Tujuan pembuatan aplikasi tersebut adalah untuk memudahkan *user* dalam memesan tiket tanpa

perlu mengantri saat membeli tiket yang diinginkan. Cukup dengan melalui aplikasi tersebut *user* dapat melakukan pembelian berbagai macam tiket, khususnya *entertainment.*

Dalam aplikasi LockYourTicket terdapat berbagai macam menu yang tersedia, dimulai dari fitur *login* dan *sign up* untuk mendaftarkan *user* dan memvalidasi akun *user* agar bisa masuk ke aplikasi LockYourTicket. Pada *home* menu, *user* dapat melihat rekomendasi tiket, lokasi terkini, serta kupon yang dapat digunakannya. Pada *menu search, user* dapat melakukan pencarian yang diinginkan serta memiliki banyak kategori yang disediakan pada menu tersebut *user* dapat langsung melakukan pembelian tiket. Pada *menu 'My Events', user* dapat melihat *history* terkait pemesanan tiketnya serta tiket yang akan datang. Pada *main menu* yang terakhir, yaitu *profile*, *user* dapat melihat data diri pada akunnya serta menu menu informasi lain seperti notifikasi, metode pembayaran, bahasa, dan menu yang dapat mengubah peraturan.

Pada aplikasi LockYourTicket, *database* yang digunakan ada 2, yaitu *database* yang berbasis 'phpMyAdmin' dan yang kedua, yaitu *database* lokal yang berasal dari aplikasi Android Studio, yaitu *SQLite Database*.