

# **LAPORAN PROYEK AKHIR**

## **IMPLEMENTASI PEMBUATAN APLIKASI TRAVEL**

### **“EZGO” BERBASIS MOBILE APP**



#### **Disusun oleh:**

Karin Eldora	NIM: 00000068097
Kenny B. Lawson	NIM: 00000081065
Jantzen Fernandes	NIM: 00000074907

**Mata Kuliah:** (IS534-A) Mobile Application Development

**Dosen Pengampu:** Ahmad Faza, S.Kom., M.T.I. dan  
Monika Evelin Johan, S.Kom., M.M.S.I.

**PROGRAM STUDI SISTEM INFORMASI**  
**FAKULTAS TEKNIK DAN INFORMATIKA**  
**UNIVERSITAS MULTIMEDIA NUSANTARA**

**TANGERANG, 2024**

## HALAMAN PENGESAHAN LAPORAN

Dokumen ini telah disetujui sebagai laporan perancangan aplikasi untuk penggerjaan proyek akhir program studi Sistem Informasi, khususnya dalam mata kuliah Mobile Application Development. Aplikasi ini berjudul "EZGO" yang bertujuan untuk memberikan pengalaman perjalanan terbaik bagi pengguna melalui berbagai fitur seperti pembelian tiket, pemesanan hotel, paket tour, dan eksplorasi tempat destinasi. EZGO dirancang untuk meningkatkan kenyamanan dan kemudahan dalam merencanakan, memesan, dan mengelola perjalanan, serta memastikan setiap langkah dalam prosesnya terorganisir dan aman.

<b>Topik</b>	Aplikasi Mobile Layanan Travel
<b>Judul Laporan (Indonesia)</b>	Implementasi Pembuatan Aplikasi Travel "EZGO" Berbasis Mobile App
<b>Judul Laporan (Inggris)</b>	Implementation of Developing Travel Application "EZGO" Based on Mobile App
<b>Dosen Pembimbing</b>	<< 078756 - Ahmad Faza, S.Kom., M.T.I. dan 071281 - Monika Evelin Johan, S.Kom., M.M.S.I. >>

NIM	NAMA	TANDA TANGAN
00000068097	Karin Eldora	
00000081065	Kenny Budiarso Lawson	
00000074907	Jantzen Fernandes	

Tangerang, 23 Juli 2024

## KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga kami berhasil menyelesaikan laporan yang berjudul “*Implementasi Pembuatan Aplikasi Travel ‘EZGO’ Berbasis Mobile App*” tepat pada waktunya. Proyek ini merupakan bagian penting dari ujian akhir untuk mata kuliah “IS534 - Mobile Application Development” dan dirancang untuk membantu kami memahami serta menerapkan konsep-konsep dalam pengembangan aplikasi mobile, baik dari segi teori maupun praktik. Dalam proses pengembangan proyek ini, kami memanfaatkan Android Studio sebagai platform utama untuk merancang dan mengembangkan aplikasi mobile EZGO.

Penyelesaian laporan ini tidak akan tercapai tanpa dukungan dan bantuan dari berbagai pihak yang berperan penting dalam setiap tahapan proyek ini. Kami ingin menyampaikan rasa terima kasih kepada semua pihak yang telah memberikan dukungan, bimbingan, dan dorongan. Khususnya kepada:

1. Bapak Ahmad Faza, S.Kom., M.T.I. dan Ibu Monika Evelin Johan, S.Kom., M.M.S.I., selaku dosen pembimbing kami yang telah dengan sabar memberikan arahan, bimbingan, serta wawasan selama proses pembuatan proyek ini.
2. Rekan-rekan mahasiswa Program Studi Sistem Informasi yang senantiasa berjuang bersama dalam menyelesaikan tugas-tugas dan proyek, serta saling memberikan semangat dan dukungan.
3. Semua pihak yang telah memberikan dukungan, motivasi, dan bantuan yang tidak dapat kami sebutkan satu per satu. Kontribusi mereka, baik secara langsung maupun tidak langsung, telah memainkan peran penting dalam penyelesaian laporan ini.

Kami juga menghargai kritik, saran, dan masukan yang konstruktif yang telah kami terima selama proses penyusunan laporan ini. Umpan balik tersebut telah membantu kami dalam menyempurnakan laporan ini agar dapat diselesaikan dengan baik. Semoga laporan ini dapat memberikan manfaat dan wawasan yang berguna bagi para pembaca, terutama dalam konteks perancangan dan pengembangan aplikasi mobile, serta dapat menjadi referensi yang bermanfaat untuk pengembangan aplikasi di masa depan.

## DAFTAR ISI

<b>HALAMAN PENGESAHAN LAPORAN.....</b>	<b>2</b>
<b>KATA PENGANTAR.....</b>	<b>3</b>
<b>DAFTAR ISI.....</b>	<b>4</b>
<b>BAB I - PENDAHULUAN.....</b>	<b>7</b>
1.1. Latar Belakang.....	7
1.2. Rumusan Masalah.....	8
1.3. Tujuan.....	8
1.4. Profile Konsep Aplikasi EZGO.....	9
1.5. Site Diagram Aplikasi EZGO.....	11
<b>BAB II - ANALISIS DAN DESAIN SISTEM.....</b>	<b>13</b>
2.1. Identifikasi Masalah Bisnis.....	13
2.2. Proses Bisnis Aplikasi.....	14
2.3. Identifikasi Pengguna.....	16
2.4. Fitur Aplikasi.....	17
2.5. User Requirements.....	21
2.6. Desain Database / Entity Relationship Diagram (ERD).....	24
<b>BAB III - HASIL DAN PEMBAHASAN.....</b>	<b>29</b>
3.1. Layout Aplikasi (Interface).....	29
3.1.1. Splash Screen.....	29
3.1.2. Register Screen.....	30
3.1.3. Login Screen.....	31
3.1.4. Main Menu (Homepage).....	32
3.1.5. Main Menu (Fitur Pembelian Ticket).....	34
3.1.6. Main Menu (Fitur Pembelian Hotel).....	39
3.1.7. Main Menu (Fitur Pembelian Paket Tour).....	44
3.1.8. Menu Explore Destinasi Wisata.....	49
3.1.9. Menu Lihat History Pembelian (Order).....	54
3.1.10. Menu Profile.....	55
3.2. Code Aplikasi Android (Android Studio).....	58
3.2.1. AboutActivity.....	58
3.2.2. AccountInformationActivity.....	59
3.2.3. AdapterHome.....	61
3.2.4. AdapterSearch.....	63
3.2.5. AdapterViewExplore.....	65
3.2.6. AdapterViewHotel.....	67

3.2.7. AdapterViewTicket.....	69
3.2.8. AdapterViewTour.....	71
3.2.9. ChangePassActivity.....	73
3.2.10. city.....	74
3.2.11. ExploreData.....	75
3.2.12. ExploreDetailActivity.....	76
3.2.13. ExploreFragment.....	78
3.2.14. HomeFragment.....	80
3.2.15. hotel.....	83
3.2.16. HotelActivity.....	84
3.2.17. HotelData.....	87
3.2.18. HotelDetailActivity.....	88
3.2.19. HotelViewActivity.....	90
3.2.20. internalDB.....	91
3.2.21. KotaActivity.....	94
3.2.22. LandingActivity.....	96
3.2.23. LandingAdapter.....	97
3.2.24. LandingFragmentOne.....	97
3.2.25. LandingFragmentThree.....	98
3.2.26. LandingFragmentTwo.....	99
3.2.27. location.....	99
3.2.28. LoginActivity.....	100
3.2.29. MainActivity.....	102
3.2.30. MyItem.....	105
3.2.31. OrderFragment.....	107
3.2.32. PaymentActivity.....	109
3.2.33. PaymentDetailActivity.....	112
3.2.34. PaymentSuccessActivity.....	116
3.2.35. ProfileFragment.....	117
3.2.36. ResponseFourObjectList.....	119
3.2.37. ResponseMessage.....	120
3.2.38. ResponseNoObject.....	120
3.2.39. ResponseOneObject.....	121
3.2.40. ResponseOneObjectList.....	121
3.2.41. ResponseThreeObjectList.....	122
3.2.42. SearchActivity.....	123
3.2.43. SearchItem.....	125
3.2.44. SecurityActivity.....	127

3.2.45. SettingsActivity.....	127
3.2.46. SingupActivity.....	128
3.2.47. ticket.....	130
3.2.48. TicketActivity.....	131
3.2.49. TicketDetailActivity.....	134
3.2.50. TicketViewActivity.....	137
3.2.51. tour.....	139
3.2.52. TourActivity.....	140
3.2.53. TourDetailActivity.....	143
3.2.54. TourViewActivity.....	144
3.2.55. User.....	146
3.2.56. WebViewActivity.....	147
3.3. Code Back-End Aplikasi (PHP).....	148
3.3.1. accountController.php.....	148
3.3.2. cityController.php.....	150
3.3.3. database.php.....	152
3.3.4. exploreController.php.....	156
3.3.5. locationController.php.....	157
3.3.6. loginController.php.....	159
3.3.7. notfound.php.....	161
3.3.8. orderController.php.....	162
3.3.9. router.php.....	175
3.3.10. searchController.php.....	176
3.3.11. transactionController.php.....	177
<b>BAB IV - PENUTUP.....</b>	<b>179</b>
4.1. Catatan Aspek Penilaian.....	179
4.2. Kesimpulan.....	184
4.3. Saran.....	185
<b>LAMPIRAN.....</b>	<b>186</b>
<b>PERAN MASING-MASING ANGGOTA.....</b>	<b>186</b>

# **BAB I**

## **PENDAHULUAN**

### **1.1. Latar Belakang**

Pengembangan aplikasi travel berbasis mobile dipandu oleh dinamika perubahan perilaku konsumen dan kemajuan teknologi. Seiring dengan semakin banyaknya pengguna smartphone, masyarakat kini mencari solusi yang memberikan akses cepat dan mudah ke informasi perjalanan serta pemesanan layanan wisata. Kemajuan teknologi, seperti pemrosesan pembayaran online dan kecerdasan buatan, memberikan landasan untuk meningkatkan pengalaman pengguna dengan menyediakan fitur-fitur yang memudahkan perencanaan perjalanan, reservasi akomodasi, dan pilihan transportasi. Peningkatan koneksi internet dan kesadaran akan teknologi modern juga memainkan peran penting dalam mengarahkan industri travel menuju solusi yang lebih canggih dan responsif terhadap kebutuhan pengguna.

Situasi persaingan yang ketat di industri travel didorong oleh persyaratan konsumen yang semakin tinggi, sehingga mendorong perusahaan untuk terus berinovasi. Aplikasi travel tidak hanya menjadi sarana untuk merencanakan perjalanan, tetapi juga platform yang menyediakan pengalaman personalisasi dengan memanfaatkan data pengguna. Oleh karena itu, pengembangan aplikasi travel berbasis mobile bukan hanya respons terhadap tuntutan konsumen, tetapi juga strategi adaptasi terhadap perubahan lingkungan global yang tidak pasti. Penyedia layanan perjalanan yang tanggap terhadap kondisi yang tidak terduga dapat menggunakan aplikasi mereka untuk memberikan pembaruan real-time kepada pengguna, seperti rekomendasi destinasi wisata yang sedang populer dan akses mudah ke pemberitahuan perjalanan pelanggan.

Melihat fenomena tersebut, kami memilih untuk mengembangkan aplikasi travel berbasis mobile bernama EZGO. Aplikasi ini bertujuan untuk memberikan pengalaman perjalanan terbaik bagi pengguna melalui berbagai fitur seperti pembelian tiket, pemesanan hotel maupun paket tour, serta eksplorasi berbagai tempat destinasi. Dengan antarmuka yang ramah pengguna, integrasi database secara real-time, dan penerapan teknologi canggih, EZGO diharapkan dapat memenuhi kebutuhan perjalanan modern dengan menyediakan solusi yang cepat, mudah, dan aman. EZGO hadir sebagai solusi untuk mengatasi berbagai permasalahan yang dihadapi oleh

pelancong, termasuk ketidakpastian harga dan ketersediaan tiket, serta ketidaknyamanan dalam merencanakan perjalanan.

Aplikasi travel EZGO bertujuan untuk memberikan solusi yang responsif terhadap kebutuhan dinamis para pengguna. Melalui aplikasi ini, proses perjalanan menjadi lebih terorganisir, praktis, dan memuaskan. Dengan demikian, EZGO tidak hanya mempermudah hidup masyarakat dalam merencanakan dan mengelola perjalanan mereka, tetapi juga meningkatkan transparansi dan optimalisasi dalam industri travel secara keseluruhan. Integrasi yang kuat dengan database memastikan keamanan dan keberlanjutan aktivitas pengguna secara mutakhir, mulai dari pembuatan akun hingga transaksi pembayaran.

## **1.2. Rumusan Masalah**

Berdasarkan pemaparan latar belakang di atas, dapat diidentifikasi beberapa masalah yang ditemukan, di antaranya adalah sebagai berikut:

1. Masyarakat mengalami kesulitan dalam mengakses informasi perjalanan dan pemesanan layanan wisata dengan cepat dan mudah melalui aplikasi mobile.
2. Ketidakpastian harga dan ketersediaan tiket sering kali menjadi kendala bagi pengguna dalam merencanakan perjalanan.
3. Perusahaan travel perlu terus berinovasi untuk memenuhi tuntutan konsumen yang semakin tinggi dan beradaptasi dengan perubahan lingkungan global yang tidak pasti.
4. Kurangnya platform yang menyediakan pengalaman personalisasi dengan memanfaatkan data pengguna untuk merekomendasikan destinasi wisata yang sedang populer.
5. Keterbatasan aplikasi travel saat ini dalam memberikan pembaruan real-time dan akses mudah ke pemberitahuan perjalanan pelanggan, yang penting untuk meningkatkan kenyamanan dan keamanan pengguna.

## **1.3. Tujuan**

Berdasarkan rumusan masalah yang telah diidentifikasi, tujuan pengembangan aplikasi EZGO adalah sebagai berikut:

1. Memudahkan masyarakat dalam mengakses informasi perjalanan dan pemesanan layanan wisata melalui aplikasi mobile yang cepat dan mudah digunakan.
2. Mengurangi ketidakpastian harga dan ketersediaan tiket dengan menyediakan informasi yang transparan dan real-time kepada pengguna.
3. Mengimplementasikan teknologi terbaru, seperti pemrosesan pembayaran online untuk mendorong inovasi dalam perusahaan travel dan memenuhi tuntutan konsumen yang semakin tinggi serta beradaptasi dengan perubahan lingkungan global yang tidak pasti.
4. Menyediakan platform yang menawarkan pengalaman personalisasi bagi pengguna dengan memanfaatkan data untuk merekomendasikan destinasi wisata yang sedang populer dan relevan.
5. Mengatasi keterbatasan aplikasi travel saat ini dengan memberikan pembaruan dan penggunaan database secara real-time, serta akses mudah ke pemberitahuan perjalanan, sehingga meningkatkan kenyamanan dan keamanan pengguna selama perjalanan.

#### **1.4. Profile Konsep Aplikasi EZGO**

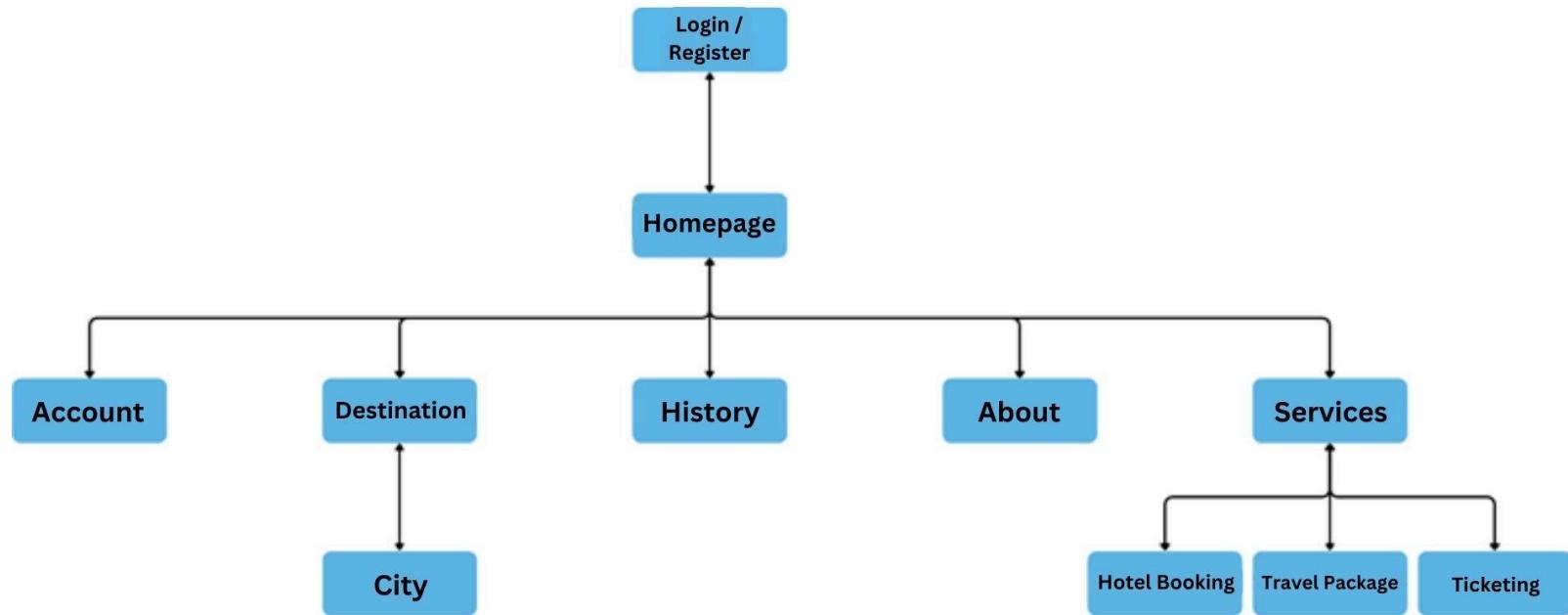
EZGO merupakan sebuah aplikasi layanan travel berbasis mobile yang dirancang untuk memberikan pengalaman perjalanan terbaik bagi pengguna. Aplikasi ini menyediakan berbagai fitur dan penawaran khusus yang memudahkan pengguna dalam merencanakan, memesan, dan mengelola perjalanan mereka. Fokus utama EZGO adalah pada pembelian tiket untuk berbagai moda transportasi, termasuk pesawat, kapal, dan kereta, yang dapat diakses dengan mudah ke seluruh Indonesia. Selain itu, aplikasi ini juga menyediakan layanan pemesanan hotel maupun paket tour, dan eksplorasi tempat destinasi, yang semuanya diintegrasikan dalam satu platform yang user-friendly.

Keunggulan utama EZGO terletak pada kemudahan pembelian tiket, hotel, atau paket tour, serta integrasinya dengan database secara real-time. Hal ini memastikan keamanan dan kesinambungan aktivitas pengguna mulai dari pembuatan akun, login, pencarian, pembelian, transaksi, hingga melihat history pembelian. Setiap langkah dalam proses perjalanan pengguna terorganisir dan tersimpan secara aman dalam sistem aplikasi. EZGO juga menonjolkan antarmuka yang ramah pengguna, dengan navigasi yang didesain untuk mempermudah pengguna dalam menjelajahi fitur-fitur yang ditawarkan. Selain itu, EZGO mengimplementasikan

teknologi terbaru seperti pemrosesan pembayaran online untuk meningkatkan pengalaman pengguna. Seluruh teknologi yang dimiliki EZGO memberikan landasan untuk menyediakan fitur-fitur yang memudahkan perencanaan perjalanan, reservasi akomodasi, dan pilihan transportasi. Dengan adanya pembaruan real-time dan akses mudah ke pemberitahuan perjalanan, EZGO mampu mengatasi berbagai permasalahan yang dihadapi oleh pelancong, termasuk ketidakpastian harga dan ketersediaan tiket. Dengan demikian, EZGO tidak hanya mempermudah hidup masyarakat dalam merencanakan dan mengelola perjalanan mereka, tetapi juga meningkatkan transparansi dan optimalisasi dalam industri travel secara keseluruhan.

Melalui aplikasi EZGO, pengguna dapat melakukan registrasi atau login, kemudian menjelajahi dan memesan berbagai layanan travel seperti tiket pesawat, kapal, kereta, hotel, dan paket tour. Pengguna dapat memasukkan informasi perjalanan yang diinginkan, memilih metode pembayaran, dan menyelesaikan transaksi secara online. Setelah pembayaran terkonfirmasi, detail perjalanan akan tersedia di aplikasi. Setelah perjalanan selesai, detail tiket pembelian akan tersimpan di riwayat (history) pengguna sehingga dapat dilihat kembali. Selain itu, pengguna juga dapat mengeksplorasi berbagai destinasi wisata, melihat informasi lengkap seperti foto dan video tempat destinasi, lokasi di Google Maps, harga tiket masuk maupun transport, serta mengelola profil akun mereka. EZGO memastikan proses pemesanan yang cepat, mudah, dan aman, dengan pembaruan real-time dan berbagai metode pembayaran yang memudahkan transaksi.

## 1.5. Site Diagram Aplikasi EZGO



Site diagram adalah representasi visual dari arsitektur informasi sebuah aplikasi yang menunjukkan koneksi antar halaman dan konten di aplikasi tersebut. Dalam kasus ini, diagram ini menjelaskan arsitektur aplikasi mobile EZGO. Dari halaman utama, terdapat beberapa halaman yang terhubung dalam aplikasi. Dan halaman-halaman dalam halaman utama juga terhubung ke halaman-halaman lainnya, berikut penjelasannya:

- [Login/Register](#): Di halaman Login/Register, pengguna dapat mendaftar atau masuk ke akun mereka untuk mengakses aplikasi mobile EZGO.
- [Homepage](#): Home Page adalah halaman pertama yang akan dilihat pengguna setelah masuk ke akun mereka. Di halaman ini, pengguna disajikan berbagai opsi rekomendasi destinasi wisata dan layanan lainnya yang tersedia di aplikasi mobile EZGO. Pengguna dapat menuju halaman account, halaman destination, halaman history, halaman about, dan halaman services.
- [Account](#): Di halaman akun, pengguna dapat melihat informasi pribadi mereka, seperti edit profile, change password, delete account, maupun log out. Pengguna juga dapat mengatur dan memperbarui informasi pribadi mereka sesuai kebutuhan.

- Destination: Di halaman destinasi, pengguna dapat melihat daftar destinasi populer yang direkomendasikan. Pengguna juga dapat membaca deskripsi dan detail lainnya seperti foto, video, atau maps dari destinasi tersebut.
- History: Di halaman riwayat (history), pengguna dapat melihat semua transaksi dan pemesanan yang telah mereka lakukan. Halaman ini memungkinkan pengguna untuk mengakses detail perjalanan sebelumnya dan mengelola informasi pemesanan mereka.
- About: Di halaman about, pengguna dapat melihat informasi tentang aplikasi mobile EZGO, seperti latar belakang EZGO, keunggulan EZGO, dan informasi staf.
- Service: Halaman service digunakan oleh pengguna untuk memesan layanan dari aplikasi travel EZGO. Aplikasi EZGO menyediakan tiga produk kepada pengguna, yaitu pemesanan tiket (pesawat, kereta, dan kapal), pemesanan hotel, dan pemesanan paket wisata tour. Pengguna dapat melihat beberapa produk yang dijual EZGO dan memesannya di sana. Jika stok produk habis, aplikasi akan memberi tahu pengguna bahwa produk tersebut terjual. Namun jika ada stok, maka pengguna dapat melanjutkan ke konfirmasi pembelian. Setelah pengguna memilih apa yang ingin mereka pesan, akan ada formulir reservasi untuk mengkonfirmasi pemesanan pengguna.

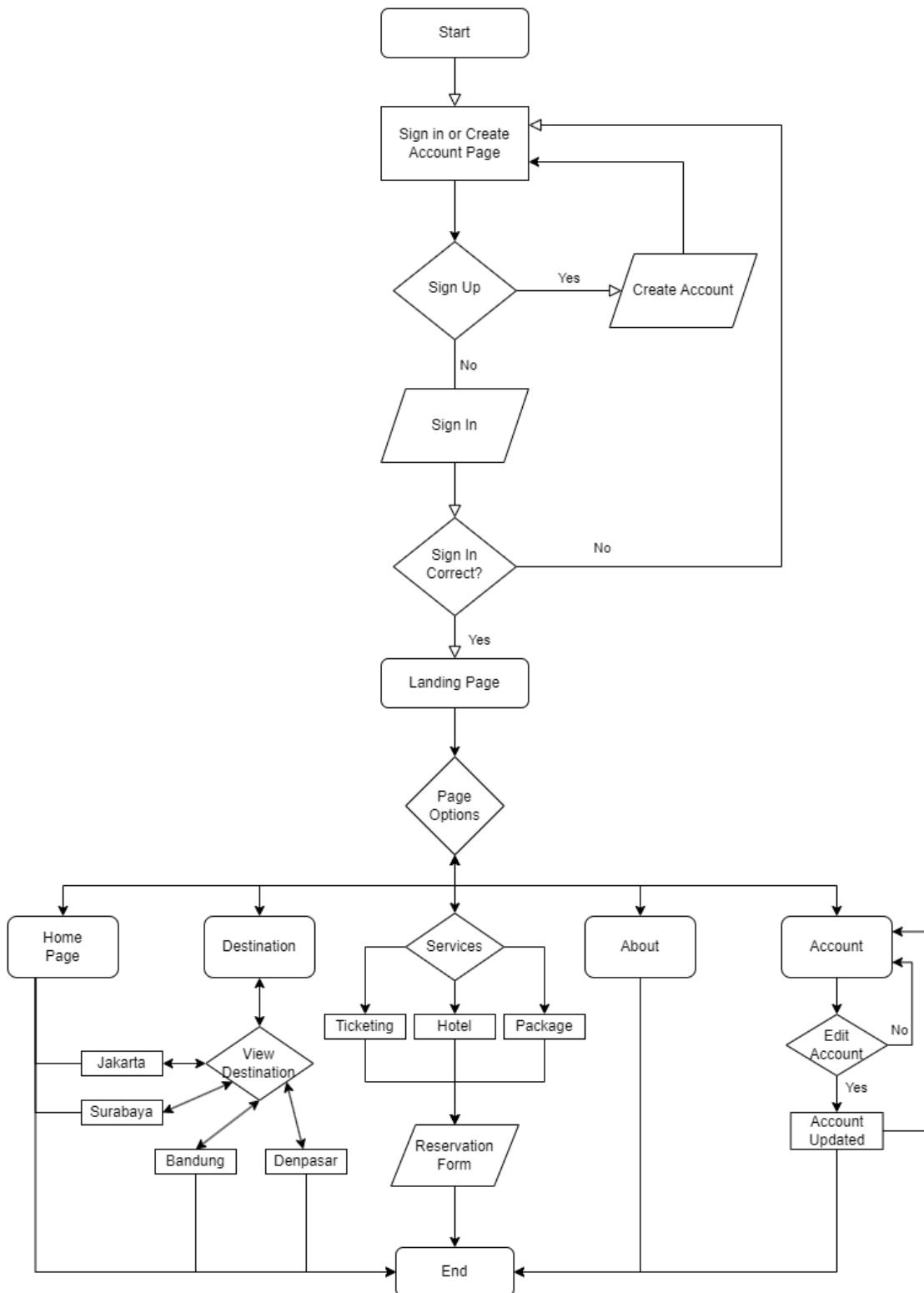
## **BAB II**

### **ANALISIS DAN DESAIN SISTEM**

#### **2.1. Identifikasi Masalah Bisnis**

Dalam industri travel yang semakin berkembang, terdapat sejumlah masalah bisnis yang harus diatasi untuk memastikan kelancaran layanan dan kepuasan pelanggan. Salah satu masalah utama yang dihadapi adalah kesulitan dalam mengakses informasi perjalanan dan pemesanan layanan wisata dengan cepat dan mudah. Pengguna seringkali menghadapi ketidakpastian harga dan ketersediaan tiket, yang dapat menghambat proses perencanaan perjalanan mereka. Selain itu, ada ketidakjelasan dalam proses pembayaran yang mungkin membuat beberapa pelanggan mengalami kesulitan. Persaingan yang ketat di industri travel mendorong perusahaan untuk terus berinovasi guna memenuhi tuntutan konsumen yang semakin tinggi dan beradaptasi dengan perubahan lingkungan global yang tidak pasti. Ketidakmampuan platform travel saat ini untuk menyediakan pengalaman personalisasi yang memanfaatkan data pengguna secara efektif juga menjadi tantangan. Banyak aplikasi travel yang belum mampu memberikan pembaruan real-time dan akses mudah ke pemberitahuan perjalanan, yang sangat penting untuk meningkatkan kenyamanan dan keamanan pengguna selama perjalanan. Pengalaman pelanggan juga perlu diperbaiki, terutama dalam hal antarmuka pengguna yang kurang responsif atau tidak intuitif. Dalam usaha mengatasi masalah-masalah ini, aplikasi EZGO hadir sebagai solusi yang inovatif dan user-friendly untuk memastikan pengalaman perjalanan yang lebih baik bagi pengguna dan meningkatkan efisiensi dalam industri travel. EZGO menawarkan integrasi database secara real-time, penggunaan teknologi terbaru seperti pemrosesan pembayaran online, dan antarmuka yang ramah pengguna untuk menyediakan layanan yang cepat, mudah, dan aman bagi para pelancong.

## 2.2. Proses Bisnis Aplikasi



Flowchart Proses Bisnis Aplikasi EZGO

Proses bisnis aplikasi travel EZGO dimulai dengan pelanggan mengunduh dan menginstal aplikasi. Setelah aplikasi berhasil terunduh dengan sempurna, pengguna akan melihat tampilan untuk melakukan registrasi atau login. Jika belum memiliki akun, pengguna harus melakukan registrasi terlebih dahulu dengan mengisi data diri yang diperlukan, seperti username, nama, kata sandi, alamat email, dan nomor telepon. Namun jika pengguna sudah memiliki akun, maka pengguna dapat langsung melakukan login dengan menggunakan username dan passwordnya. Setelah login berhasil, pengguna akan masuk ke halaman utama (homepage) aplikasi EZGO, yang menyediakan berbagai pilihan produk seperti pemesanan tiket, hotel, paket tour, fitur eksplorasi tempat destinasi, fitur melihat history pemesanan, dan pengelolaan akun pengguna. Pengguna juga dapat menjelajahi berbagai destinasi wisata di Indonesia dan melihat produk populer yang sering dibeli dan disukai oleh pengguna lain.

- Proses Pemesanan Tiket:

Untuk melakukan pemesanan tiket, pelanggan harus mengisi formulir pemesanan (Reservation Form) dengan memasukkan informasi perjalanan yang diinginkan, seperti jenis tiket, tanggal keberangkatan, tanggal kedatangan, kota asal, kota tujuan, dan jumlah penumpang. Setelah informasi lengkap diisi, pelanggan akan memilih metode pembayaran yang tersedia. Pembayaran dapat dilakukan dengan kartu kredit, kartu debit, atau transfer bank. Setelah pembayaran berhasil, pelanggan akan menerima bukti pemesanan yang tersimpan dalam aplikasi dan dapat diakses kapan saja.

- Proses Pemesanan Hotel:

Proses pemesanan hotel serupa dengan proses pemesanan tiket. Pelanggan harus mengisi formulir pemesanan (Reservation Form) dengan memasukkan informasi perjalanan yang diinginkan, seperti jenis hotel, tanggal check-in, tanggal check-out, kota tujuan, dan jumlah kamar. Setelah itu, pelanggan memilih metode pembayaran yang tersedia. Pembayaran dapat dilakukan dengan kartu kredit, kartu debit, atau transfer bank. Setelah pembayaran berhasil, pelanggan akan menerima bukti pemesanan hotel.

- Proses Pemesanan Paket Tour:

Untuk pemesanan wisata tour, pelanggan juga harus mengisi formulir pemesanan (Reservation Form) dengan memasukkan informasi perjalanan seperti jenis tour yang diinginkan, tanggal keberangkatan, dan jumlah peserta. Setelah itu, pelanggan memilih metode pembayaran

yang tersedia dan melakukan pembayaran. Setelah pembayaran berhasil, pelanggan akan menerima bukti pemesanan tour.

Selain ketiga layanan yang tersedia, juga ada fitur eksplorasi tempat destinasi yang memungkinkan pelanggan untuk melihat informasi mengenai tempat wisata dan lokasi-lokasi tertentu dari destinasi yang diinginkan. Informasi yang tersedia mencakup foto dan nama destinasi wisata, video yang dapat diakses langsung menggunakan Youtube, deskripsi destinasi wisata, lokasi yang dapat diakses langsung menggunakan Google Maps, likes dari tempat wisata tersebut. Fitur ini memudahkan pelanggan untuk merencanakan perjalanan dengan lebih baik dan mengetahui lebih banyak tentang destinasi yang akan mereka kunjungi.

Pengguna EZGO juga dapat melakukan pengaturan akun pengguna pada halaman Account(Profile). Pada halaman ini pengguna dapat mengubah informasi akunnya, mulai dari informasi Username, Nama, Email, Nomor telepon, dan Password akun. Setelah pengguna melakukan perubahan informasi akun, maka informasi akun pengguna tersebut akan otomatis di-update atau diperbaharui pada database user EZGO.

Antarmuka aplikasi EZGO dirancang untuk ramah pengguna dengan navigasi yang mudah dipahami. Pengguna dapat melakukan berbagai aktivitas seperti melihat dan mengedit profil, mengubah password, melihat riwayat pembelian tiket, menghapus akun, dan melakukan logout. Selain itu, terdapat juga fitur pencarian (search bar) yang dapat digunakan oleh pengguna untuk mencari destinasi wisata yang ingin dilihat. Dengan proses bisnis yang jelas dan fitur-fitur yang bermanfaat, aplikasi travel EZGO dapat memberikan layanan pemesanan perjalanan yang aman dan nyaman bagi pelanggan.

### **2.3. Identifikasi Pengguna**

Aplikasi travel EZGO dirancang untuk memenuhi kebutuhan berbagai kelompok pengguna yang mencari solusi praktis dengan fitur terintegrasi dalam merencanakan perjalanan melalui perangkat Android mereka. Sasaran utama aplikasi ini adalah para pelancong individu yang menginginkan kenyamanan dan efisiensi dalam mengatur perjalanan mereka, baik untuk tujuan bisnis maupun rekreasi. Selain itu, aplikasi ini juga menargetkan pengguna yang menginginkan pengalaman perjalanan yang personal dan disesuaikan dengan preferensi

masing-masing. Dengan menyediakan antarmuka yang intuitif dan fitur yang dapat disesuaikan, aplikasi ini dapat diakses oleh berbagai kalangan, termasuk pengguna dengan tingkat pengalaman perjalanan yang beragam, mulai dari yang baru pertama kali hingga yang sudah berpengalaman. Pendekatan ini memastikan bahwa aplikasi EZGO dapat menjadi teman setia bagi para pelancong dengan memberikan pengalaman perjalanan yang lebih lancar dan memuaskan.

Lebih lanjut, aplikasi EZGO juga dirancang untuk menjangkau pengguna dari berbagai segmen demografis dan geografis. Aplikasi ini tidak hanya bermanfaat bagi pelancong domestik, tetapi juga bagi wisatawan internasional yang membutuhkan kemudahan dalam mengakses informasi perjalanan dan pemesanan layanan wisata di Indonesia. Selain itu, EZGO juga memfasilitasi pengguna yang sering bepergian dengan menyediakan fitur-fitur yang membantu mereka mengatur jadwal perjalanan dengan lebih mudah. Dengan adanya fitur pembaruan real-time, pengguna dapat selalu mendapatkan informasi terbaru tentang perjalanan mereka, termasuk perubahan jadwal dan rekomendasi destinasi. Aplikasi ini juga memperhatikan kebutuhan pengguna dengan preferensi khusus, seperti wisata kuliner, wisata alam, atau wisata budaya, sehingga setiap pengguna dapat merencanakan perjalanan yang sesuai dengan minat mereka. Dengan demikian, EZGO berkomitmen untuk menyediakan solusi perjalanan yang komprehensif dan adaptif, yang mampu memenuhi kebutuhan dinamis berbagai kelompok pengguna dan memastikan pengalaman perjalanan yang lebih terorganisir, praktis, dan memuaskan.

## **2.4. Fitur Aplikasi**

Fitur atau fungsi yang disediakan dalam aplikasi EZGO adalah sebagai berikut:

### **1. Register**

Fitur register memungkinkan pengguna yang belum memiliki akun EZGO untuk mendaftarkan diri. Pengguna akan diminta untuk mengisi data diri seperti username, nama, kata sandi, alamat email, dan nomor telepon. Data yang telah diisi akan tersimpan dalam database aplikasi, sehingga pengguna dapat melanjutkan ke tahap login di kemudian hari.

## 2. Login

Fitur login digunakan oleh pengguna yang telah mendaftar untuk mengakses akun mereka. Pengguna hanya perlu memasukkan username dan kata sandi yang telah terdaftar. Fitur ini memastikan bahwa hanya pengguna terdaftar yang dapat mengakses layanan dan informasi pribadi mereka di dalam aplikasi.

## 3. Menu Homepage

Menu homepage adalah tampilan utama setelah pengguna berhasil login. Di sini, pengguna dapat melihat berbagai pilihan produk seperti tiket pesawat, hotel, dan paket tour. Selain itu, pengguna juga dapat mengeksplorasi berbagai destinasi wisata di Indonesia serta melihat produk-produk rekomendasi yang terpopuler dan paling sering dibeli oleh pengguna lain.

## 4. Layanan Pembelian Tiket

Fitur layanan pembelian tiket memungkinkan pengguna untuk mencari dan membeli tiket pesawat sesuai kebutuhan mereka. Pengguna dapat memasukkan informasi perjalanan seperti tanggal keberangkatan, tujuan, dan jumlah penumpang. Setelah pencarian, tiket yang tersedia akan ditampilkan dan pengguna dapat memilih serta melakukan pembayaran langsung melalui aplikasi.

## 5. Layanan Pemesanan Hotel

Fitur layanan pemesanan hotel memungkinkan pengguna untuk mencari dan memesan kamar hotel sesuai kebutuhan mereka. Pengguna dapat mengisi informasi seperti kota tujuan, tanggal check-in dan check-out, jenis kamar, serta jumlah kamar yang dibutuhkan. Aplikasi akan menampilkan hotel yang tersedia beserta detailnya, dan pengguna dapat melakukan pemesanan dan pembayaran secara online.

## 6. Layanan Pemesanan Paket Tour

Fitur layanan pemesanan paket tour menyediakan berbagai pilihan tour yang dapat dipesan oleh pengguna. Pengguna dapat mengisi informasi seperti kota tujuan, tanggal tour, dan jumlah peserta. Aplikasi akan menampilkan tour yang tersedia beserta detail itinerary, dan pengguna dapat memilih serta melakukan pembayaran langsung melalui aplikasi.

## **7. Filter Cari Tiket, Hotel, dan Paket Tour Sesuai Preferensi Pengguna**

Fitur ini memungkinkan pengguna untuk memfilter hasil pencarian tiket, hotel, dan paket tour sesuai preferensi mereka. Pengguna dapat menyaring hasil berdasarkan harga, lokasi, tanggal, dan fasilitas yang diinginkan, sehingga memudahkan mereka menemukan pilihan yang paling sesuai dengan kebutuhan dan anggaran.

## **8. Edit dan Update Pesanan**

Fitur ini memungkinkan pengguna untuk mengedit dan memperbarui pesanan mereka. Misalnya, pengguna dapat mengubah pesanan mereka dengan menambahkan jumlah kursi untuk tiket pesawat, mengatur ulang jumlah malam menginap di hotel, atau mengubah jumlah peserta dalam paket tour. Seluruh pesanan yang diubah akan diperbarui langsung, termasuk harga dan ketersediaan layanan. Dengan fitur ini, pengguna dapat dengan mudah menyesuaikan pesanan mereka sebelum melakukan pembayaran akhir.

## **9. Layanan Eksplorasi Destinasi Wisata**

Fitur layanan eksplorasi destinasi wisata memungkinkan pengguna untuk mencari dan menemukan informasi tentang berbagai destinasi wisata di Indonesia. Pengguna dapat melihat detail destinasi, termasuk foto dan video tempat destinasi, lokasi di Google Maps, harga tiket masuk maupun transport, deskripsi, dan likes dari destinasi wisata tersebut. Selain itu, layanan ini juga dilengkapi dengan search bar yang memungkinkan pengguna untuk melakukan pencarian destinasi wisata dengan filter tertentu. Misalnya, jika pengguna mencari destinasi wisata dengan kata kunci “National”, maka hanya produk-produk yang mengandung kata “National” yang akan ditampilkan. Fitur ini membantu pengguna dalam merencanakan perjalanan wisata mereka dengan lebih baik, cepat, dan lebih spesifik sesuai preferensi mereka.

## **10. Rekomendasi Destinasi Wisata dan Layanan Terpopuler**

Fitur rekomendasi menampilkan destinasi wisata yang paling populer di masyarakat sesuai kondisi terkini serta layanan yang paling sering dibeli oleh pengguna EZGO. Fitur ini memastikan bahwa pengguna selalu mendapatkan saran terbaik yang sesuai dengan preferensi dan keinginan mereka.

#### **11. Fitur Cek Lokasi Menggunakan Google Maps**

Fitur ini memungkinkan pengguna untuk melihat lokasi destinasi wisata atau tempat tour menggunakan Google Maps. Pengguna dapat dengan mudah menemukan lokasi yang mereka cari dan mendapatkan petunjuk arah yang akurat untuk sampai ke sana.

#### **12. Fitur Menampilkan Video Destinasi Wisata Menggunakan Youtube**

Fitur ini menampilkan video dari destinasi wisata yang diambil dari YouTube. Pengguna dapat melihat video untuk mendapatkan gambaran lebih jelas tentang tempat yang ingin mereka kunjungi. Fitur ini memberikan pengalaman visual yang lebih kaya dibandingkan hanya melihat foto.

#### **13. Fitur Likes Destinasi Wisata dan Layanan yang Tersedia**

Pengguna dapat memberikan likes pada destinasi wisata dan layanan yang mereka sukai. Fitur ini membantu aplikasi dalam mengumpulkan data tentang preferensi pengguna dan memberikan rekomendasi yang lebih tepat. Selain itu, pengguna lain dapat melihat destinasi atau layanan yang populer berdasarkan jumlah likes.

#### **14. Layanan History Transaksi Pembelian**

Fitur ini menampilkan riwayat transaksi pembelian pengguna, termasuk tiket, hotel, dan paket tour yang telah dibeli. Pengguna dapat melihat detail setiap transaksi, termasuk tanggal, jumlah, dan status pembayaran. Fitur ini membantu pengguna untuk melacak dan mengelola pengeluaran mereka dengan lebih baik.

#### **15. Search Bar**

Fitur search bar pada aplikasi EZGO memungkinkan pengguna untuk mencari destinasi wisata, hotel, tiket pesawat, dan paket tour dengan cepat dan mudah. Pengguna cukup mengetik kata kunci atau frasa tertentu untuk menemukan informasi yang mereka butuhkan. Selain itu, search bar ini dilengkapi dengan opsi filter yang membantu pengguna menyaring hasil pencarian sesuai preferensi mereka. Misalnya, ketika pengguna mencari destinasi wisata dengan kata kunci "National", maka hanya produk-produk yang mengandung kata "National" yang akan ditampilkan.

## **16. Menu Profile Pengguna**

Menu profil pengguna menampilkan informasi pribadi seperti username, nama, password, email, dan nomor telepon. Pengguna dapat melakukan berbagai aktivitas seperti mengedit profil, mengubah password, melihat informasi akun, serta mengakses halaman "About". Di halaman "About", pengguna dapat melihat informasi tentang aplikasi EZGO, seperti latar belakang, keunggulan, dan informasi staf. Pengguna juga memiliki opsi untuk menghapus akun dan logout. Fitur ini memastikan bahwa pengguna dapat mengelola informasi pribadi mereka dengan mudah dan aman.

## **17. Detail Layanan yang Tersedia dan Detail Pesanan**

Fitur ini menyediakan informasi lengkap mengenai layanan yang tersedia dan detail pesanan yang telah dibuat oleh pengguna. Setiap layanan, baik itu tiket, hotel, atau paket tour, dilengkapi dengan deskripsi, harga, fasilitas, kuantitas, tanggal, dan lokasi. Pengguna dapat melihat semua detail ini sebelum melakukan pemesanan.

## **18. Metode Pembayaran Online**

EZGO menyediakan berbagai metode pembayaran online yang aman, termasuk kartu kredit, kartu debit, dan transfer bank. Bank yang tersedia juga beragam seperti BCA, Mandiri, BNI, dan BRI. Pengguna dapat memilih metode pembayaran yang paling sesuai dengan preferensi mereka. Fitur ini memastikan bahwa transaksi dapat dilakukan dengan mudah dan cepat.

## **19. Delete dan Logout Akun**

Fitur ini memungkinkan pengguna untuk menghapus akun mereka atau logout dari aplikasi. Menghapus akun akan menghapus semua data pengguna dari database aplikasi. Fitur logout memungkinkan pengguna untuk keluar dari akun mereka dengan aman setelah selesai menggunakan aplikasi.

## **2.5. User Requirements**

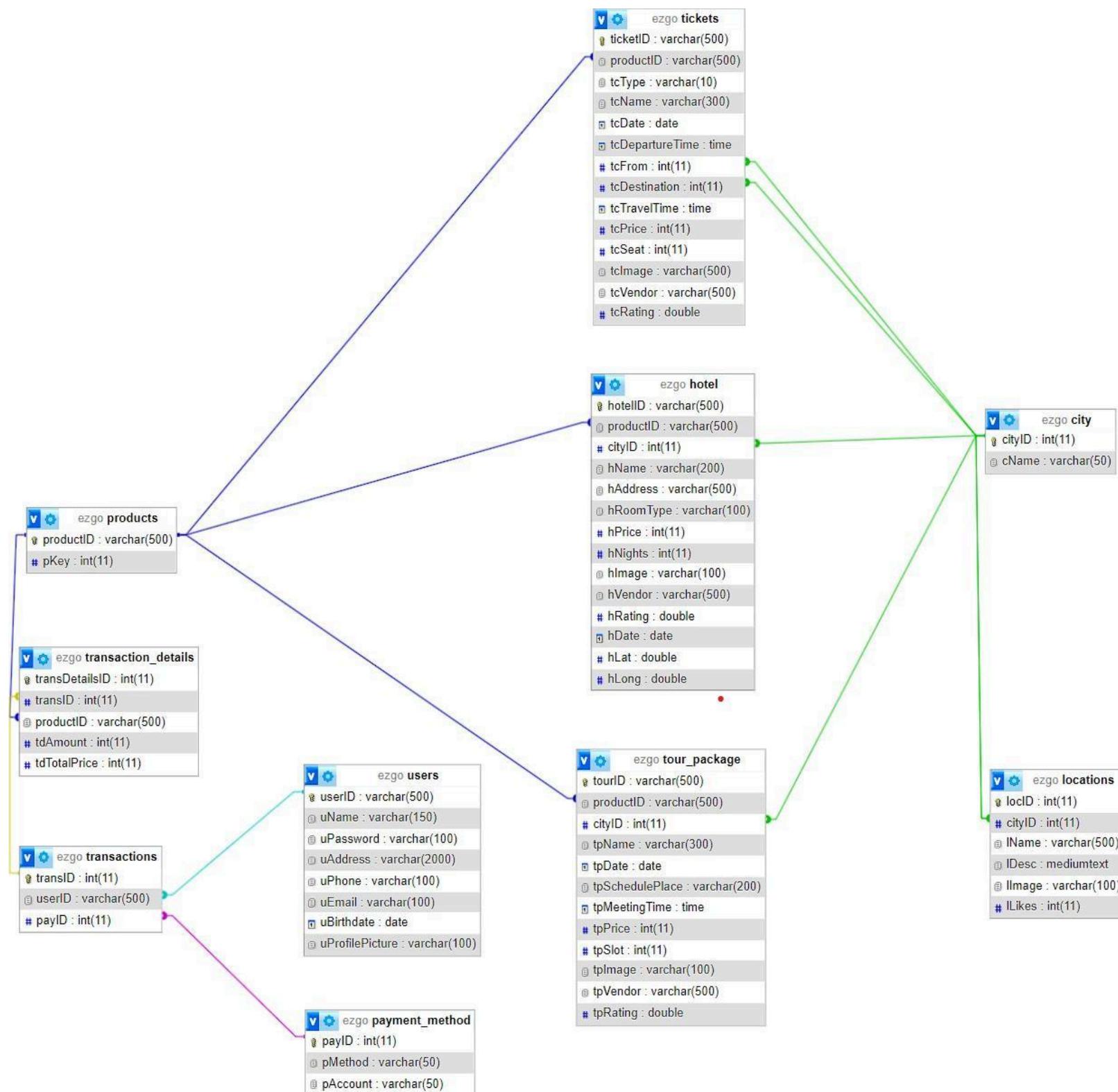
User requirements adalah deskripsi spesifik mengenai kebutuhan dan ekspektasi pengguna terhadap aplikasi EZGO. Tujuan dari spesifikasi ini adalah memastikan aplikasi mampu menyediakan fitur dan fungsi yang dibutuhkan untuk memberikan pengalaman

perjalanan yang optimal dan memuaskan. Dengan mendetailkan kebutuhan pengguna, pengembang dapat merancang aplikasi yang intuitif, aman, dan mudah digunakan, serta mampu merespons perubahan dinamis dalam industri travel. Berikut adalah daftar user requirements untuk aplikasi EZGO:

1. Melakukan login menggunakan username dan password yang telah mereka daftarkan sebelumnya.
2. Menyimpan informasi seperti username, nama, password, alamat email, dan nomor telepon.
3. Mengelola profil pengguna, termasuk menyimpan, mengedit, mengubah, menghapus, dan memperbarui informasi pribadi.
4. Mendapatkan rekomendasi destinasi wisata yang populer berdasarkan data pengguna dan kondisi terkini.
5. Melihat berbagai destinasi wisata yang direkomendasikan pada menu Homepage dan melihat lokasi akuratnya lewat Google Maps.
6. Menyediakan informasi yang jelas tentang destinasi wisata, termasuk deskripsi, harga, gambar, video, likes, dan lokasi.
7. Melihat video destinasi wisata yang terintegrasi dengan YouTube.
8. Melakukan eksplorasi destinasi wisata, hotel, tiket pesawat, dan paket tour dengan cepat dan mudah menggunakan fitur search bar yang dilengkapi dengan opsi filter.
9. Memilih dan memesan tiket untuk berbagai moda transportasi seperti pesawat, kapal, dan kereta.
10. Memilih dan memesan hotel dengan memasukkan informasi seperti kota tujuan, tanggal check-in dan check-out, serta jenis kamar.
11. Memilih dan memesan paket tour dengan memasukkan informasi seperti kota tujuan, tanggal keberangkatan, dan jumlah peserta.
12. Melakukan filter pencarian untuk tiket, hotel, dan paket tour sesuai dengan preferensi pengguna.
13. Menambahkan tiket, hotel, atau paket tour yang telah dipilih ke keranjang belanja.
14. Mengubah jumlah pesanan atau membatalkan pesanan dari transaksi belanja.
15. Menghitung total biaya pesanan berdasarkan item di halaman transaksi belanja.

16. Melakukan konfirmasi penerimaan pesanan yang mencakup rincian pesanan, biaya total, dan detail perjalanan.
17. Melakukan pemesanan layanan perjalanan.
18. Mendukung metode pembayaran seperti kartu kredit, kartu debit, dan transfer bank.
19. Melihat riwayat transaksi pembelian tiket, hotel, dan paket tour yang telah dilakukan sebelumnya.
20. Memberikan likes pada destinasi wisata dan layanan yang tersedia.
21. Menerima pembaruan real-time dan pemberitahuan perjalanan, termasuk perubahan jadwal dan rekomendasi destinasi.
22. Mengelola akun pengguna, termasuk fitur seperti mengubah password, menghapus akun, dan logout dari aplikasi.

## 2.6. Desain Database / Entity Relationship Diagram (ERD)



Entity-Relationship Diagram (ERD) di atas menggambarkan struktur database aplikasi EZGO, yang terdiri dari 10 tabel dengan relasi yang saling terkait. Berikut adalah penjelasan detail terkait setiap tabel dan relasinya:

### 1. Tabel products

Tabel products merupakan tabel induk yang menyimpan informasi mengenai semua produk yang ditawarkan oleh EZGO, termasuk tiket, hotel, dan paket tour. Tabel ini terdiri dari kolom *productID* (ID produk) dan *pKey* (kata kunci produk).

### 2. Tabel tickets

Tabel tickets menyimpan informasi detail mengenai tiket perjalanan yang tersedia, seperti pesawat, kapal, dan kereta. Tabel ini terdiri dari kolom *ticketID* (ID unik untuk setiap tiket), *productID* (ID produk terkait yang merujuk ke tabel products), *tcType* (jenis tiket, misalnya pesawat, kapal, atau kereta), *tcName* (nama tiket atau nama perjalanan), *tcDate* (tanggal perjalanan), *tcDepartureTime* (waktu keberangkatan), *tcFrom* (kota asal), *tcDestination* (kota tujuan), *tcTravelTime* (lama perjalanan), *tcPrice* (harga tiket), *tcSeat* (jumlah kursi yang tersedia), *tcImage* (gambar tiket), *tcVendor* (nama vendor atau operator perjalanan), *tcRating* (rating tiket berdasarkan ulasan pengguna).

Relasi:

- ONE-TO-ONE dengan *products* (satu tiket hanya dapat dimiliki oleh satu produk)
- MANY-TO-ONE dengan *city* (banyak tiket dapat terkait dengan satu kota)

### 3. Tabel hotel

Tabel hotel menyimpan informasi mengenai hotel yang tersedia untuk dipesan oleh pengguna. Tabel ini terdiri dari kolom *hotelID* (ID unik untuk setiap hotel), *productID* (ID produk terkait yang merujuk ke tabel products), *cityID* (ID kota terkait yang merujuk ke tabel *city*), *hName* (nama hotel), *hAddress* (alamat hotel), *hRoomType* (jenis kamar yang tersedia di hotel), *hPrice* (harga per malam), *hNights* (jumlah malam menginap), *hImage* (gambar hotel), *hVendor* (nama vendor atau operator hotel), *hRating* (rating hotel berdasarkan ulasan pengguna), *hDate* (tanggal pemesanan atau menginap), *hLat* (latitude lokasi hotel), *hLong* (longitude lokasi hotel).

Relasi:

- ONE-TO-ONE dengan *products* (satu hotel hanya dapat dimiliki oleh satu produk)

- MANY-TO-ONE dengan *city* (banyak hotel dapat berada di satu kota)

#### 4. Tabel tour\_package

Tabel *tour\_package* menyimpan informasi mengenai paket wisata tour yang ditawarkan, termasuk deskripsi dan harga. Tabel ini terdiri dari kolom *tourID* (ID unik untuk setiap paket tour), *productID* (ID produk terkait yang merujuk ke tabel *products*), *cityID* (ID kota terkait yang merujuk ke tabel *city*), *tpName* (nama paket tour), *tpDate* (tanggal paket tour), *tpSchedulePlace* (tempat jadwal dalam paket tour), *tpMeetingTime* (waktu pertemuan untuk memulai paket tour), *tpPrice* (harga paket tour), *tpSlot* (jumlah slot atau tempat yang tersedia dalam paket tour), *tpImage* (gambar destinasi tempat tour), *tpVendor* (nama vendor atau penyedia paket tour), *tpRating* (rating paket tour berdasarkan ulasan pengguna).

Relasi:

- ONE-TO-ONE dengan *products* (satu paket wisata hanya dapat dimiliki oleh satu produk)
- MANY-TO-ONE dengan *city* (banyak paket wisata dapat terkait dengan satu kota)
- ONE-TO-MANY dengan *locations* (satu paket wisata dapat mengunjungi banyak lokasi)

#### 5. Tabel city

Tabel *city* menyimpan informasi mengenai kota-kota yang terkait dengan tiket, hotel, dan paket wisata. Tabel ini terdiri dari kolom *cityID* (ID kota) dan *cName* (nama kota).

Relasi:

- ONE-TO-MANY dengan *hotel* (satu kota dapat memiliki banyak hotel)
- ONE-TO-MANY dengan *tickets* (satu kota dapat memiliki banyak tiket)
- ONE-TO-MANY dengan *tour\_package* (satu kota dapat memiliki banyak paket wisata)

#### 6. Tabel locations

Tabel *locations* menyimpan informasi mengenai lokasi-lokasi wisata yang dapat dikunjungi dalam paket tour. Tabel ini terdiri dari kolom *locID* (ID unik untuk setiap lokasi wisata), *cityID* (ID kota terkait yang merujuk ke tabel *city*), *lName* (nama lokasi wisata), *lDesc* (deskripsi lokasi wisata), *lImage* (gambar atau foto lokasi wisata), *lLikes* (jumlah suka atau likes yang diterima lokasi wisata).

Relasi:

- MANY-TO-ONE dengan *city* (banyak lokasi dapat berada di satu kota)

- ONE-TO-MANY dengan *tour\_package* (satu lokasi dapat terkait dengan banyak paket wisata)
- ONE-TO-MANY dengan *users* (satu lokasi dapat dikunjungi oleh banyak pengguna)

#### 7. Tabel users

Tabel users menyimpan informasi mengenai pengguna aplikasi EZGO, termasuk data login dan kontak. Tabel ini terdiri dari kolom *userID* (ID unik untuk setiap pengguna), *uName* (nama pengguna), *uPassword* (kata sandi pengguna), *uAddress* (alamat pengguna), *uPhone* (nomor telepon pengguna), *uEmail* (alamat email pengguna), *uBirthdate* (tanggal lahir pengguna), *uProfilePicture* (foto profil pengguna).

Relasi:

- ONE-TO-MANY dengan *locations* (satu pengguna dapat mengunjungi banyak lokasi)
- ONE-TO-ONE dengan *transactions* (satu pengguna dapat melakukan satu transaksi)

#### 8. Tabel transactions

Tabel transactions menyimpan informasi mengenai transaksi yang dilakukan oleh pengguna. Tabel ini terdiri dari kolom *transID* (ID unik untuk setiap transaksi), *userID* (ID pengguna terkait yang merujuk ke tabel users), *payID* (ID metode pembayaran yang merujuk ke tabel *payment\_method*).

Relasi:

- ONE-TO-MANY dengan *products* (satu transaksi dapat memiliki banyak produk)
- ONE-TO-ONE dengan *users* (satu transaksi hanya dapat dilakukan oleh satu pengguna)
- ONE-TO-ONE dengan *payment\_method* (satu transaksi menggunakan satu metode pembayaran)

#### 9. Tabel transaction\_details

Tabel *transaction\_details* menyimpan informasi detail mengenai item yang dibeli dalam setiap transaksi. Tabel ini terdiri dari kolom *transDetailsID* (ID unik untuk setiap detail transaksi), *transID* (ID transaksi terkait yang merujuk ke tabel *transactions*), *productID* (ID produk terkait yang merujuk ke tabel *products*), *tdAmount* (jumlah item yang dibeli), *tdTotalPrice* (total harga untuk item tersebut).

Relasi:

- MANY-TO-ONE dengan *transactions* (banyak detail transaksi dapat terkait dengan satu transaksi)
- MANY-TO-ONE dengan *products* (banyak detail transaksi dapat terkait dengan satu produk)

#### *10. Tabel payment method*

Tabel *payment\_method* menyimpan informasi mengenai metode pembayaran yang tersedia di aplikasi. Tabel ini terdiri dari kolom *payID* (ID unik untuk setiap metode pembayaran), *pMethod* (nama metode pembayaran, misalnya kartu kredit, kartu debit, transfer bank), *pAccount* (akun pembayaran yang terkait dengan metode tersebut).

Relasi:

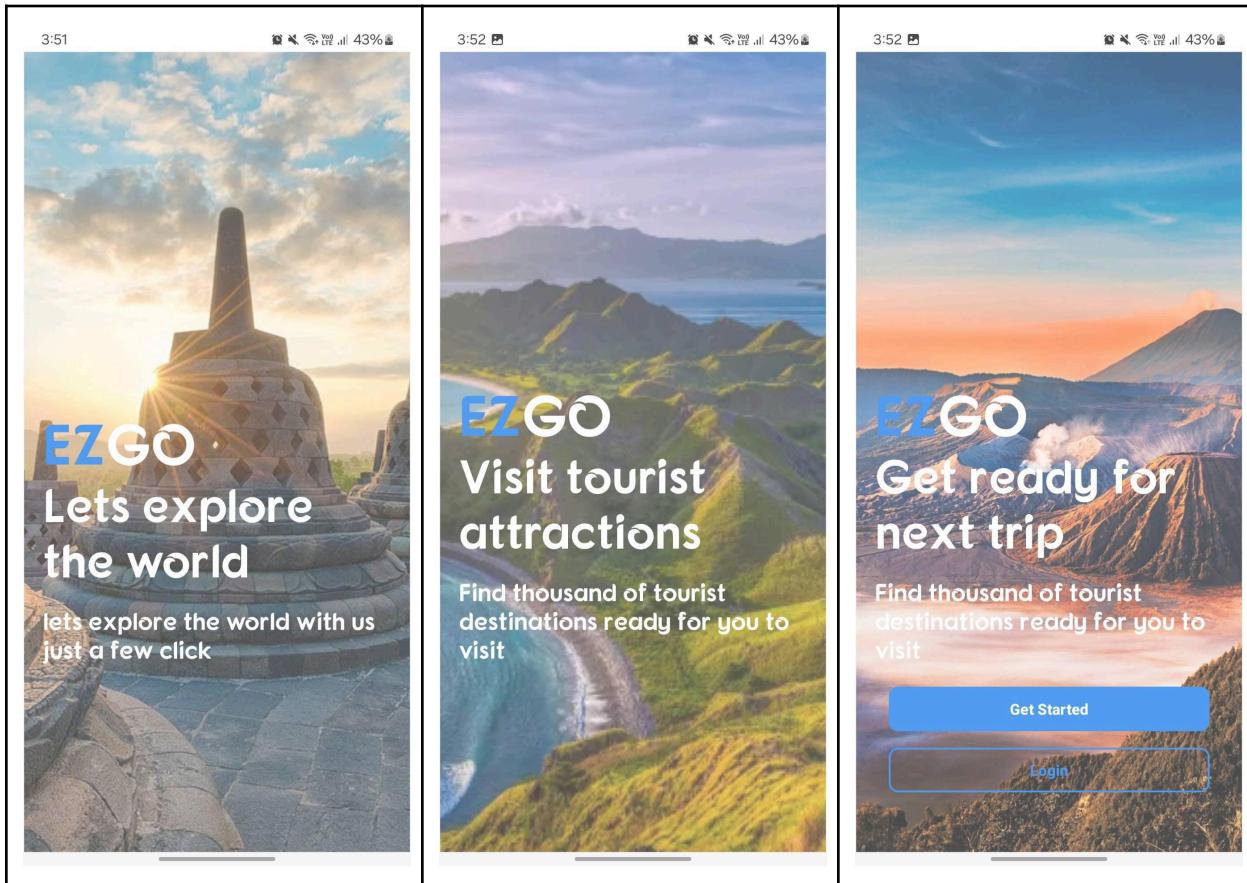
- ONE-TO-ONE dengan *transactions* (satu metode pembayaran digunakan oleh satu transaksi)

## BAB III

### HASIL DAN PEMBAHASAN

#### 3.1. Layout Aplikasi (Interface)

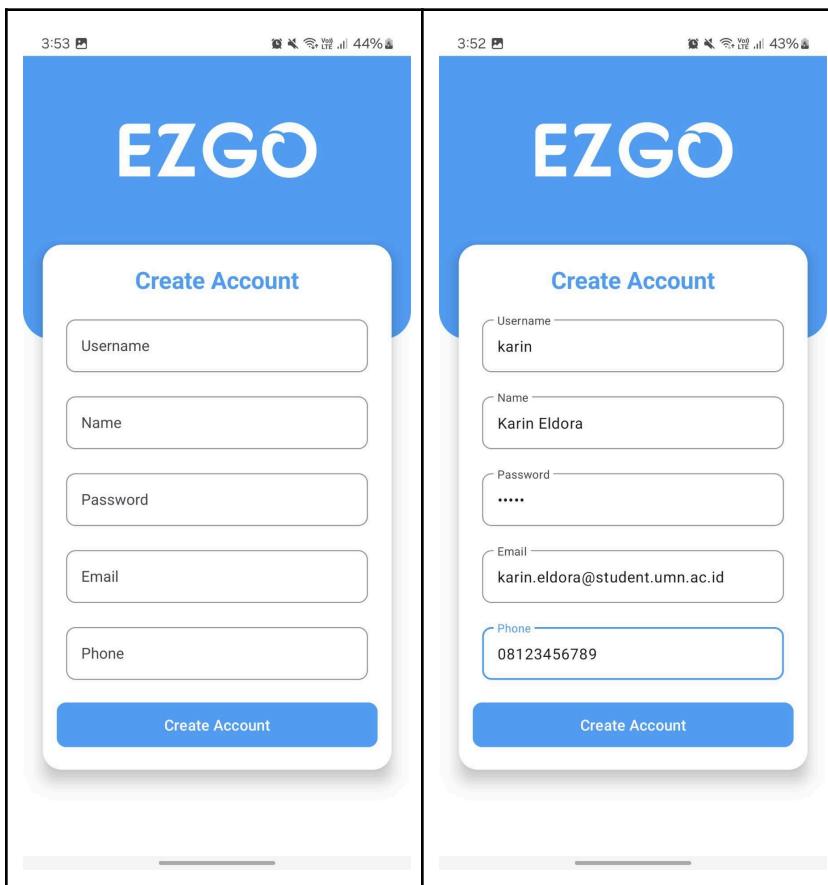
##### 3.1.1. Splash Screen



Tampilan UI *Splash Screen* EZGO yang ditunjukkan pada tiga gambar di atas menggambarkan pengalaman awal ketika pengguna pertama kali mengakses aplikasi. Tampilan ini menerapkan fitur *View Pager*, memungkinkan pengguna untuk menggeser layar ke kanan atau kiri untuk melihat berbagai halaman. Pada splash screen pertama, EZGO menampilkan *gambar candi* dengan *teks* “*Let's explore the world*” yang mengundang pengguna untuk menjelajahi dunia dengan beberapa klik saja. Pada splash screen kedua, EZGO menampilkan *gambar pemandangan indah* dengan *teks* “*Visit tourist attractions*” yang mengajak pengguna untuk menemukan ribuan destinasi wisata yang siap dikunjungi. Pada splash screen ketiga, EZGO

menunjukkan *latar belakang* gunung dengan teks “*Get ready for next trip*” memberikan semangat kepada pengguna untuk merencanakan perjalanan berikutnya dan dilengkapi dengan tombol “*Get Started*” dan “*Login*” untuk memulai eksplorasi atau masuk ke akun yang sudah ada. Pada tampilan awal ini, pengguna dapat mendaftar jika belum memiliki akun EZGO atau melakukan login jika sudah memiliki akun, memastikan akses yang mudah dan intuitif ke fitur-fitur aplikasi.

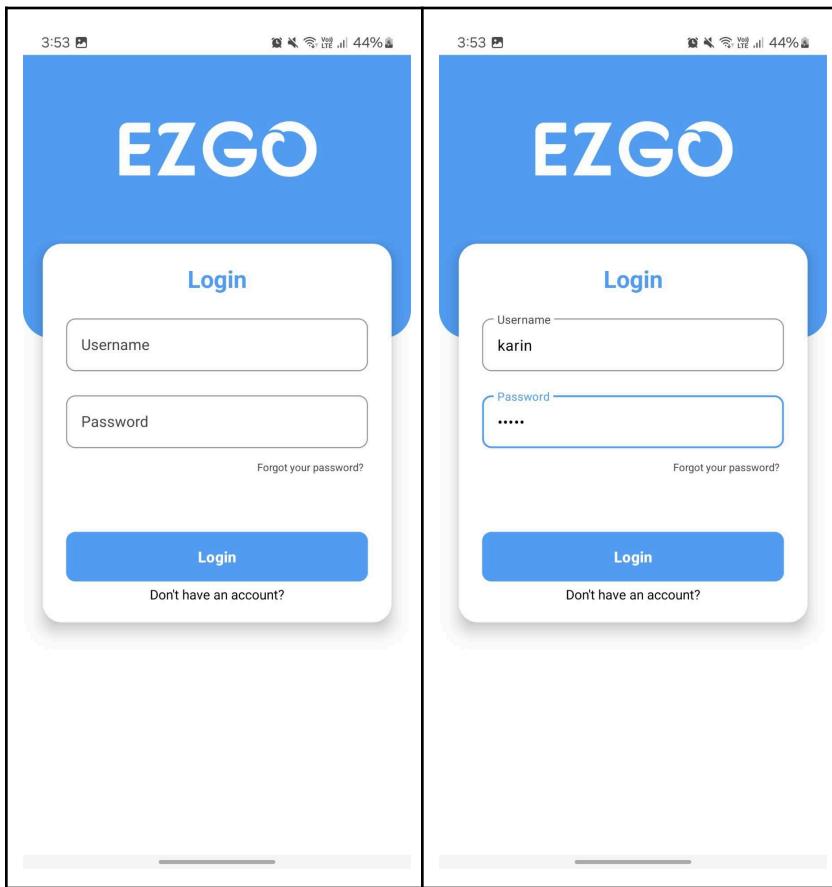
### **3.1.2. Register Screen**



Tampilan UI *Register Screen* EZGO yang ditunjukkan pada dua gambar di atas menampilkan proses pendaftaran (*register*). Pada gambar pertama, pengguna diharuskan untuk membuat akun dengan mengisi beberapa informasi yang diperlukan seperti *username*, *nama*, *password*, *email*, dan *nomor telepon*. Formulir pendaftaran ini dirancang untuk mengumpulkan data pengguna yang akan digunakan untuk membuat akun mereka di sistem EZGO. Tampilan

kedua memperlihatkan form yang telah diisi dengan contoh data. Setelah mengisi semua field, pengguna dapat menekan tombol “Create Account” untuk menyelesaikan proses pendaftaran. Setelah berhasil mendaftar, data pengguna akan tersimpan di sistem dan pengguna dapat melanjutkan untuk mengakses fitur-fitur yang disediakan oleh aplikasi EZGO. Langkah-langkah ini memastikan bahwa setiap pengguna memiliki akun yang aman dan data diri yang tersimpan dengan baik.

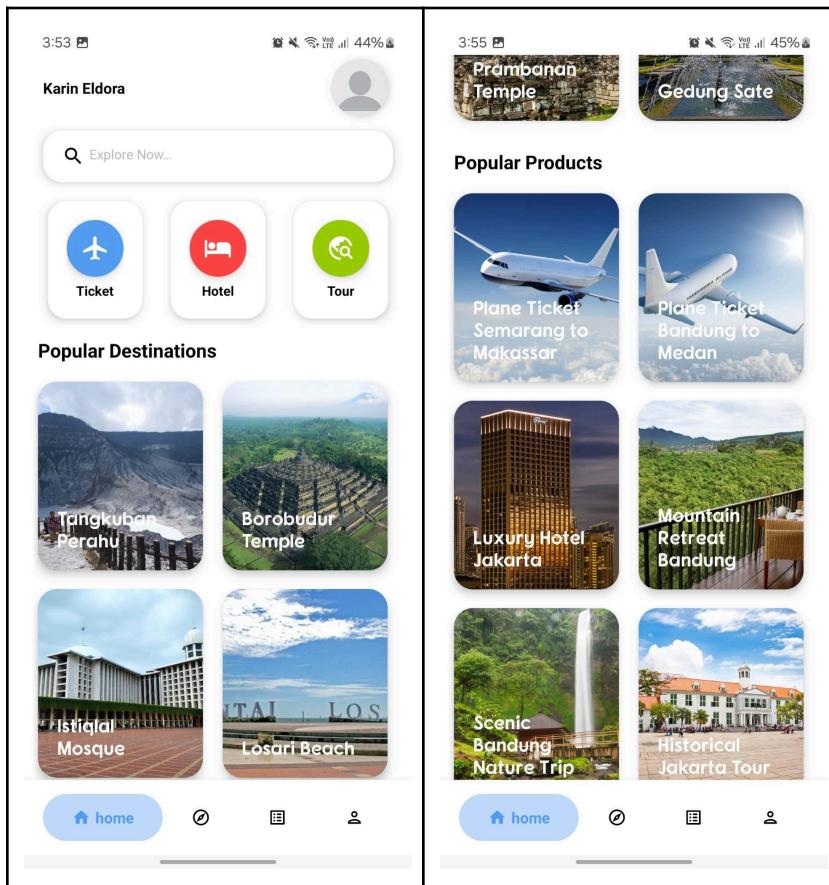
### **3.1.3. Login Screen**



Tampilan UI *Login Screen* EZGO yang ditunjukkan pada dua gambar di atas menunjukkan proses *login* yang harus dilalui pengguna setelah mereka membuat akun. Pada gambar pertama, pengguna dihadapkan pada *form login* yang meminta mereka untuk memasukkan *username* dan *password*. Terdapat juga opsi “*Forgot your password?*” bagi pengguna yang mungkin lupa kata sandi mereka. Tombol “*Login*” berada di bagian bawah form untuk memudahkan pengguna dalam mengakses akun mereka setelah memasukkan informasi

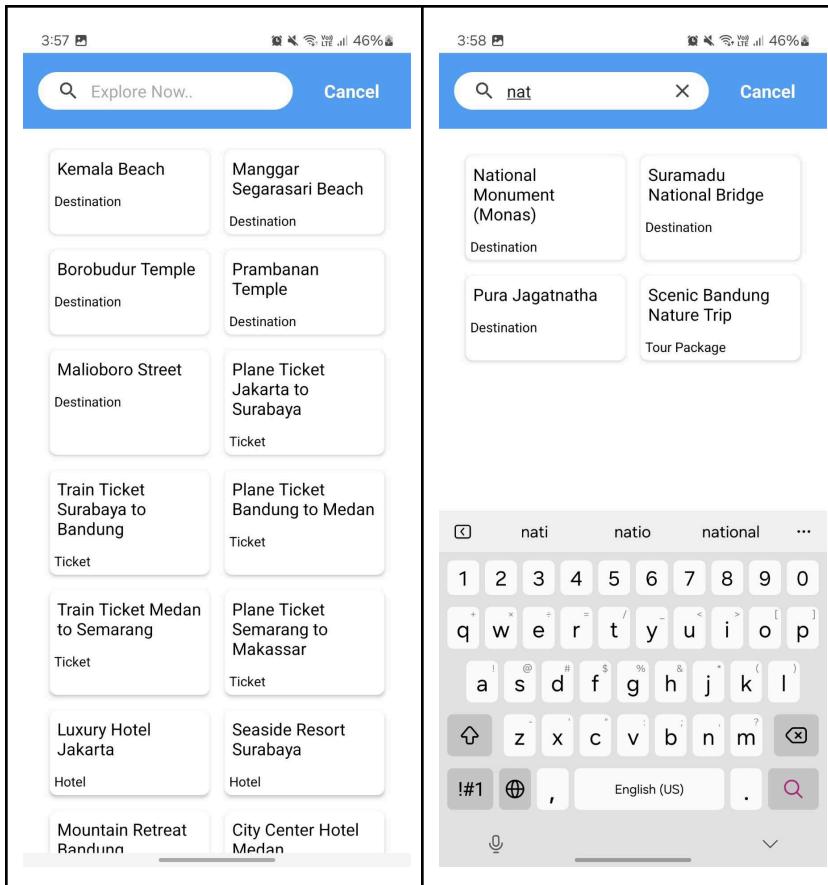
yang benar. Gambar kedua memperlihatkan form yang telah diisi dengan contoh data, dimana *username* diisi dengan “*karin*” dan *password* dengan *simbol asterisks* untuk menjaga kerahasiaan. Setelah mengisi field login, pengguna dapat menekan tombol “*Login*” untuk masuk ke akun mereka dan mulai menggunakan aplikasi. Teks “*Don't have an account?*” yang terletak di bawah tombol login memberikan akses bagi pengguna baru untuk *register*, memastikan bahwa alur pendaftaran dan login mudah dipahami dan diikuti oleh semua pengguna.

### **3.1.4. Main Menu (Homepage)**



Pada tampilan UI *Main Menu (Homepage)* EZGO yang ditunjukkan pada dua gambar di atas, pengguna akan disambut setelah berhasil melakukan login. Pada bagian atas halaman, terlihat *nama pengguna* dan *ikon profil* yang memberikan nuansa personalisasi. Terdapat juga *kolom pencarian* yang memungkinkan pengguna untuk mencari destinasi atau produk tertentu dengan cepat. Di bawahnya, terdapat *tiga tombol utama* yang menawarkan layanan inti dari aplikasi ini: *Ticket* (untuk melakukan pemesanan tiket), *Hotel* (untuk melakukan pemesanan

hotel), dan *Tour* (untuk melakukan pemesanan paket tour). Bagian ini memudahkan pengguna untuk langsung mengakses layanan yang mereka butuhkan. Di bawah tombol utama tersebut, pengguna dapat *melihat destinasi wisata di Indonesia dan layanan produk yang paling populer*. Pengguna dapat menjelajahi lebih lanjut dengan *menggulir ke bawah* untuk menemukan produk-produk populer yang sering dibeli, seperti tiket pesawat ke berbagai kota, hotel mewah di Jakarta, retret pegunungan di Bandung, dan lainnya. Tampilan ini dirancang untuk memberikan akses cepat dan mudah ke berbagai pilihan produk dan destinasi yang menarik.

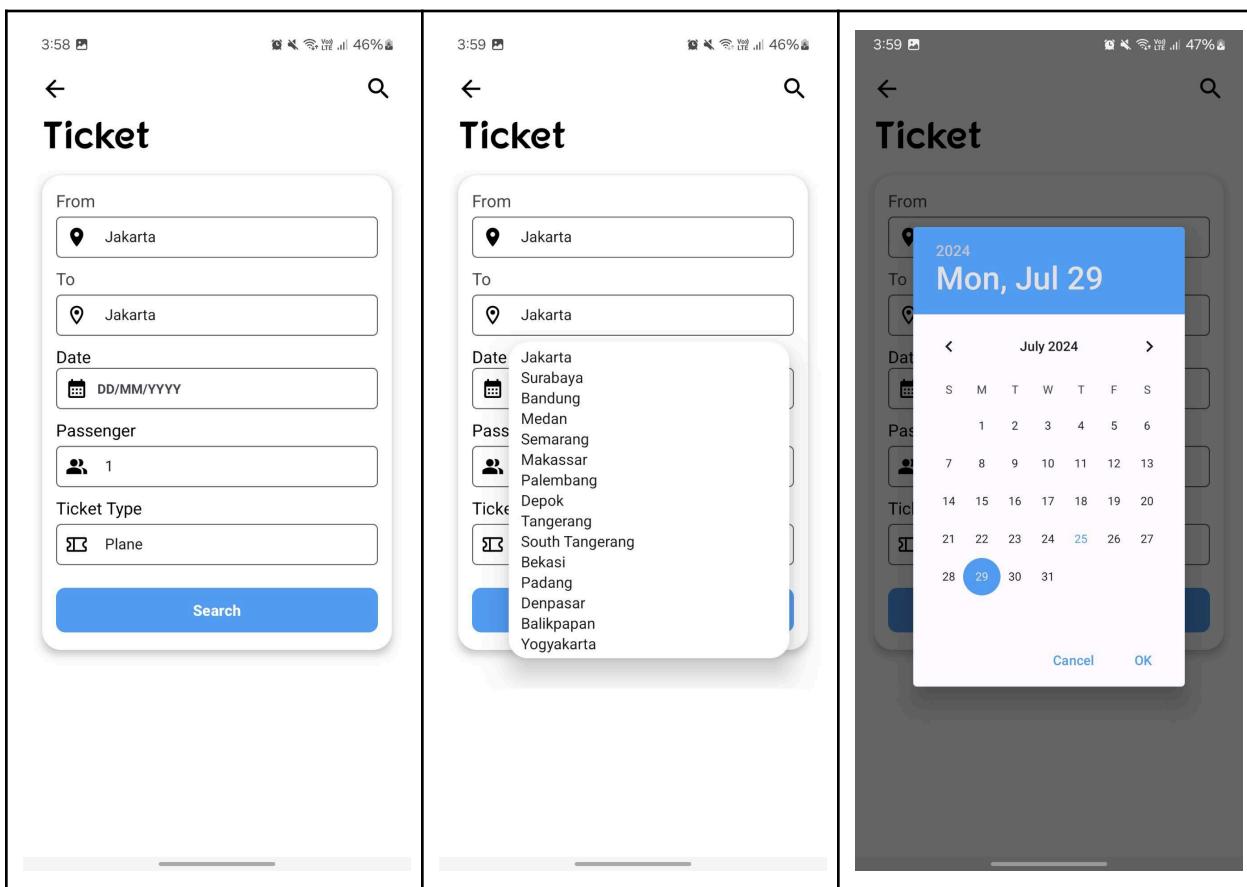


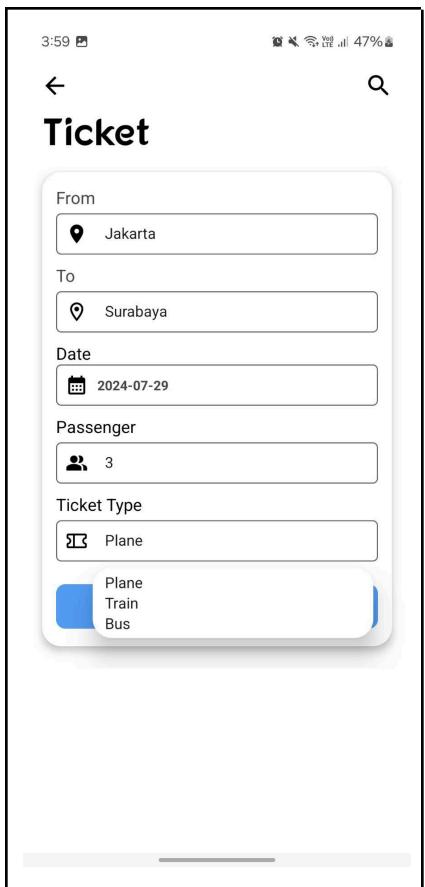
Pada tampilan UI *Main Menu (Homepage)* EZGO juga terdapat fitur pencarian “*Explore Now*” yang memungkinkan pengguna untuk mencari destinasi atau produk tertentu dengan cepat. Ketika pengguna mengklik kolom pencarian ini, *mereka akan melihat daftar lengkap destinasi, tiket, hotel, dan paket tour yang tersedia* seperti yang ditampilkan pada dua gambar di atas. Misalnya, dalam daftar yang muncul, pengguna dapat melihat berbagai destinasi populer seperti Pantai Kemala, Candi Borobudur, Jalan Malioboro, serta berbagai tiket perjalanan dan pilihan

hotel. Fitur pencarian ini juga mendukung *pencarian spesifik* dengan menggunakan *Search Bar*, sehingga ketika pengguna mulai mengetik kata kunci seperti “nat”, maka hanya produk atau destinasi yang mengandung kata tersebut yang akan ditampilkan, seperti *National* Monument (Monas), Suramadu *National* Bridge, Pura Jagatnatha, dan paket tour Scenic Bandung *Nature* Trip. Fungsi ini dirancang untuk mempermudah pengguna dalam menemukan destinasi dan produk yang mereka inginkan dengan cepat.

### **3.1.5. Main Menu (Fitur Pembelian Ticket)**

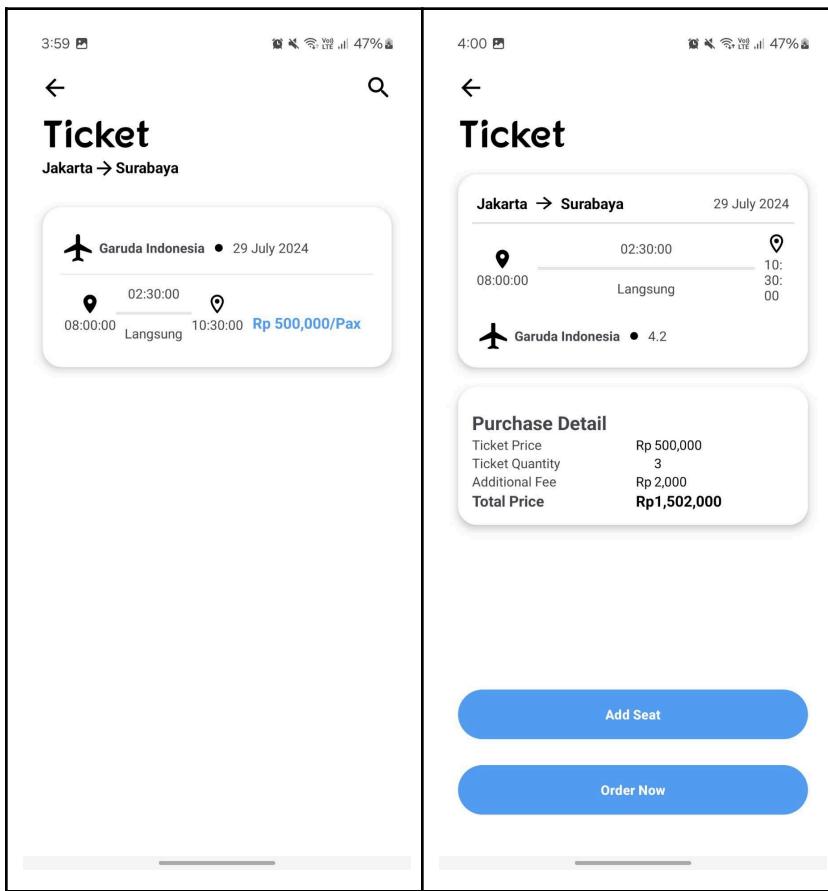
- Pengguna Mengisi Form Pencarian dan Pembelian Tiket**





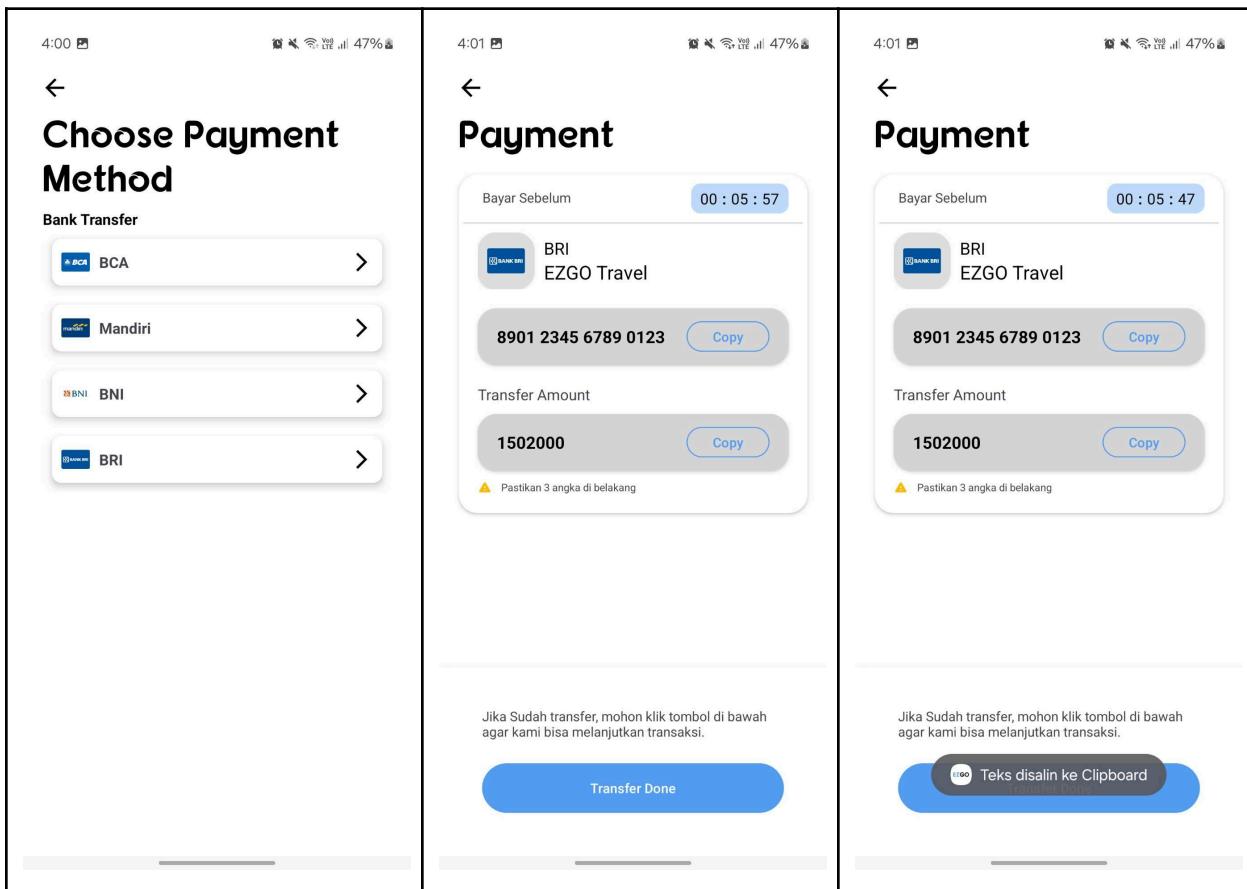
Pada tampilan UI *Main Menu (Fitur Pembelian Ticket)* EZGO khususnya pada form pencarian dan pembelian tiket seperti pada empat gambar di atas, pengguna dapat dengan mudah melakukan pencarian dan pembelian tiket sesuai kebutuhan mereka. Pada gambar pertama, pengguna diminta untuk mengisi informasi seperti kota keberangkatan (“*From*”), kota tujuan (“*To*”), tanggal keberangkatan (“*Date*”), jumlah penumpang (“*Passenger*”), dan jenis tiket (“*Ticket Type*”). Pengguna dapat memilih dari berbagai kota tujuan yang tersedia, seperti yang ditampilkan pada gambar kedua. Gambar ketiga menunjukkan pemilihan tanggal menggunakan kalender, memastikan pengguna dapat memilih tanggal keberangkatan dengan mudah. Setelah mengisi semua informasi yang diperlukan, seperti ditunjukkan pada gambar keempat dengan contoh pengisian dari Jakarta ke Surabaya, pengguna dapat menekan tombol “*Search*” untuk memfilter tiket sesuai dengan preferensi yang telah mereka masukkan. Fitur ini dirancang untuk memberikan pengalaman pemesanan tiket yang cepat dan optimal, memastikan bahwa pengguna dapat menemukan dan membeli tiket pesawat, kereta, atau bus dengan mudah sesuai kebutuhan perjalanan mereka.

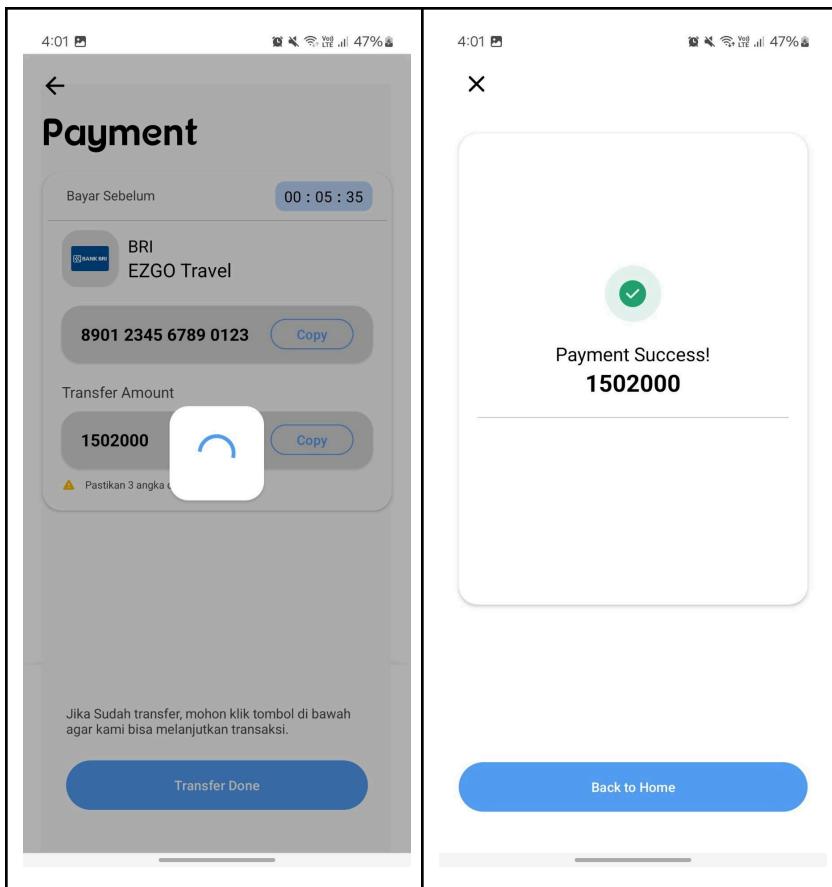
- **Ketika Pengguna Sudah Memilih Tiket**



Tampilan dua gambar UI di atas adalah tampilan *Main Menu (Fitur Pembelian Ticket)* EZGO khususnya ketika aplikasi menunjukkan beberapa pilihan tiket yang sesuai dengan input pengguna, lalu pengguna dapat memilih tiket tersebut. Setelah pengguna memasukkan informasi perjalanan dan menekan tombol “Search”, aplikasi akan menampilkan hasil pencarian tiket yang sesuai, seperti tiket pesawat Garuda Indonesia dari Jakarta ke Surabaya dengan detail penerbangan dan harga. Pengguna kemudian dapat memilih tiket yang diinginkan dan melihat halaman detail pembelian yang menampilkan rincian harga per tiket, jumlah tiket, biaya tambahan, dan total harga. Pengguna memiliki opsi untuk menambah kursi atau langsung memesan tiket dengan menekan tombol “Order Now” memastikan proses pembelian tiket yang intuitif dan transparan.

- Detail Proses Pembayaran

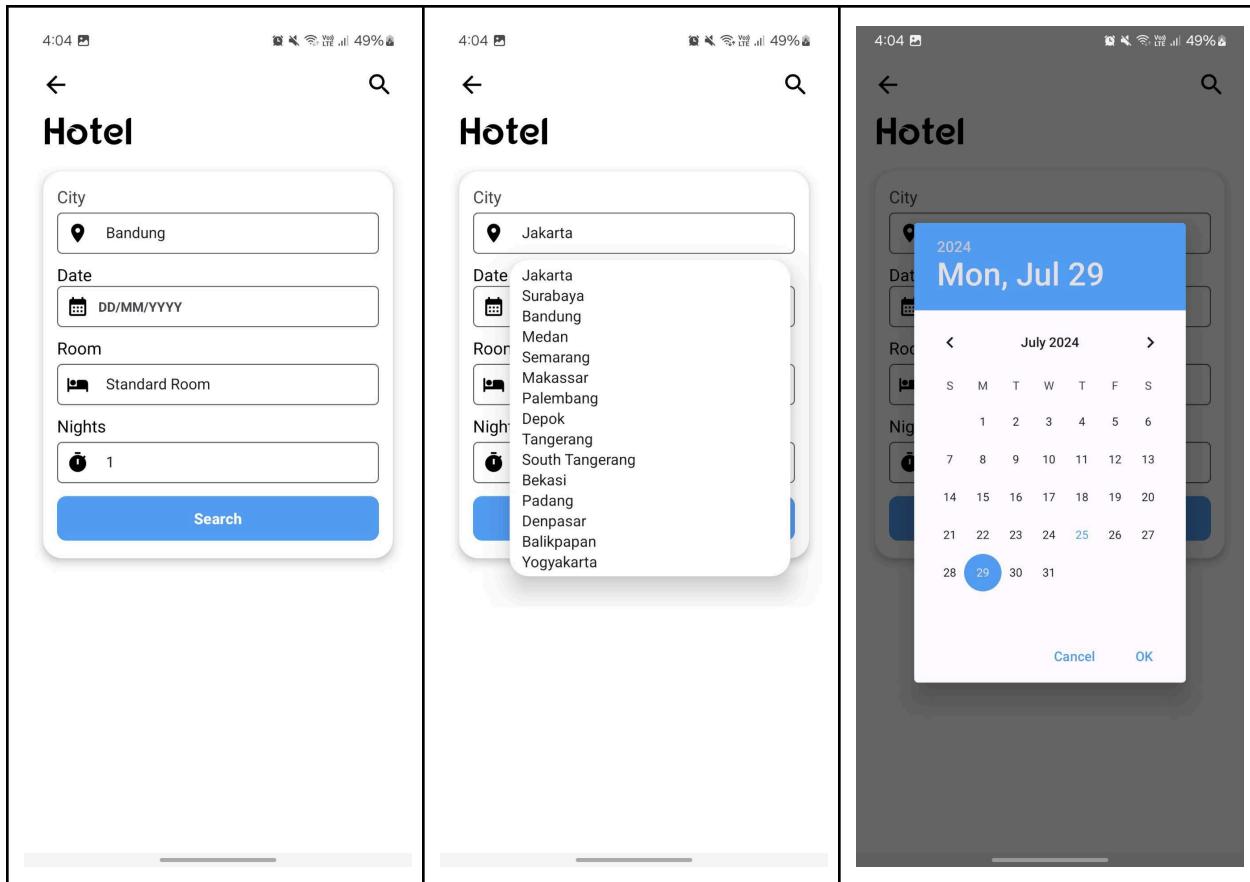


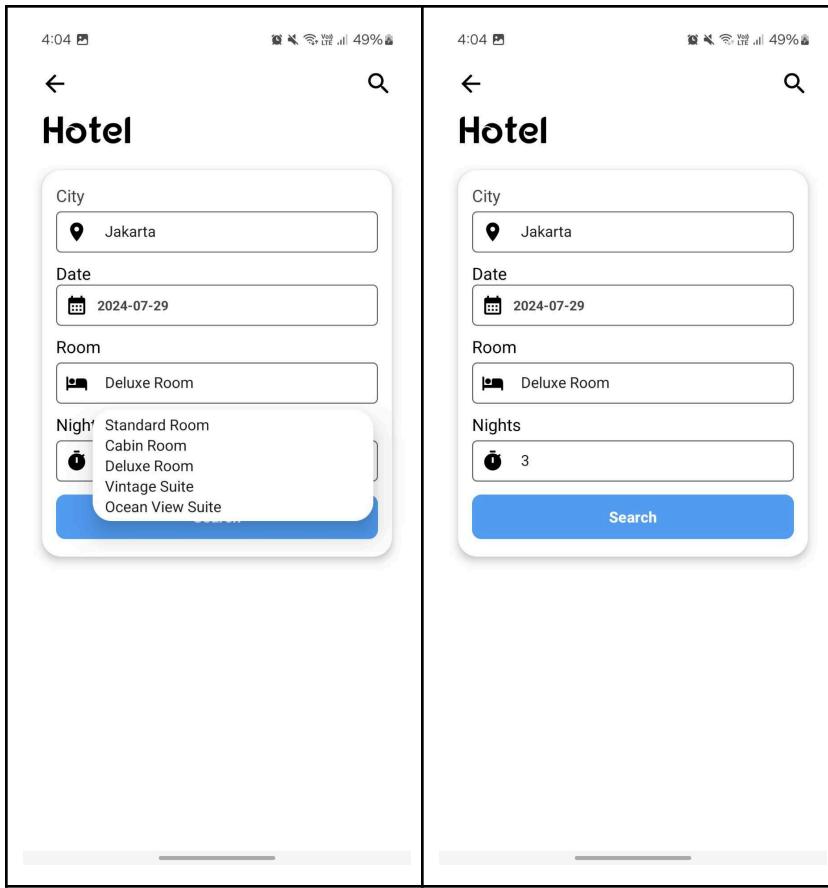


Tampilan lima gambar UI di atas adalah tampilan *Main Menu (Fitur Pembelian Ticket)* EZGO khususnya pada detail proses pembayaran. Setelah pengguna memilih tiket dan melanjutkan ke pembayaran, aplikasi akan menyediakan *berbagai metode pembayaran* untuk memudahkan transaksi. Gambar pertama menampilkan pilihan bank untuk transfer, seperti BCA, Mandiri, BNI, dan BRI. Pengguna dapat memilih salah satu bank, kemudian diarahkan ke halaman detail pembayaran yang menampilkan informasi *nomor rekening bank tujuan* dan *jumlah transfer*, lengkap dengan tombol “*Copy*” untuk memudahkan penyalinan informasi. Pada gambar ketiga, pengguna dapat melihat *informasi transfer dengan batas waktu pembayaran* yang tercantum. Setelah melakukan transfer, pengguna menekan tombol “*Transfer Done*” untuk mengkonfirmasi pembayaran. Gambar keempat menunjukkan proses verifikasi, dan seperti pada gambar kelima, pengguna akan melihat notifikasi “*Payment Success!*” yang menandakan bahwa pembayaran telah berhasil. Fitur ini memastikan bahwa pengguna dapat melakukan transaksi dengan aman dan mudah melalui berbagai pilihan metode pembayaran yang disediakan oleh EZGO.

### **3.1.6. Main Menu (Fitur Pembelian Hotel)**

- Pengguna Mengisi Form Pencarian dan Pembelian Hotel**

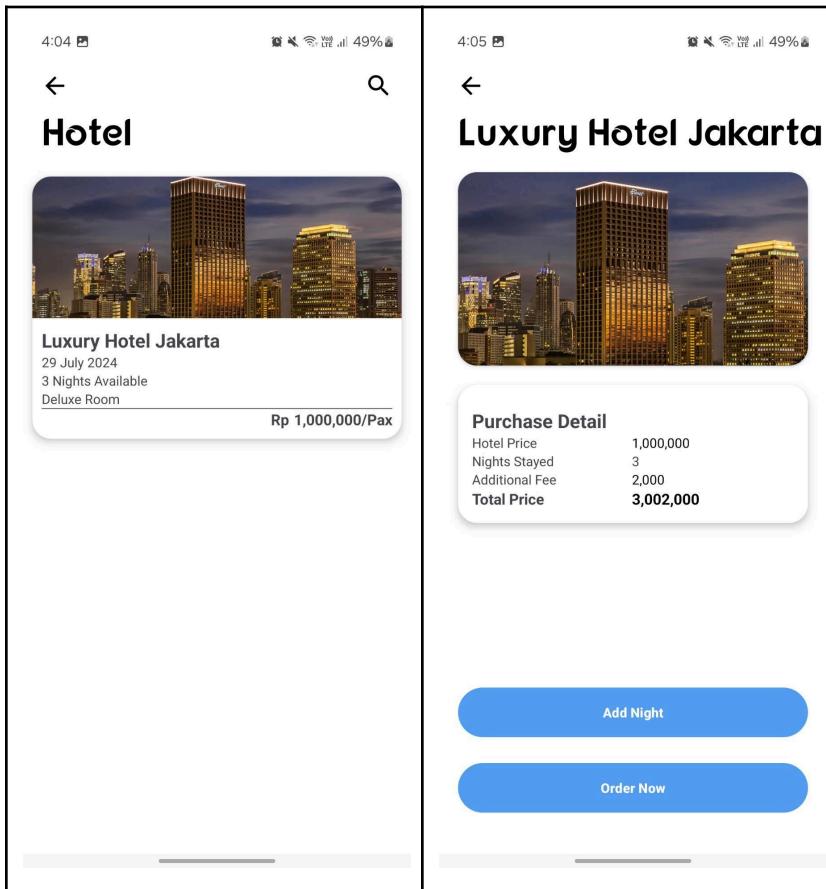




Pada tampilan UI *Main Menu (Fitur Pembelian Hotel)* EZGO khususnya pada form pencarian dan pembelian hotel seperti pada lima gambar di atas, pengguna dapat dengan mudah melakukan pencarian dan pemesanan hotel sesuai kebutuhan mereka. Pada gambar pertama, pengguna diminta untuk mengisi informasi seperti kota tujuan (“City”), tanggal check-in (“Date”), jenis kamar (“Room”), dan jumlah malam menginap (“Nights”). Pengguna dapat memilih dari berbagai kota tujuan yang tersedia, seperti yang ditampilkan pada gambar kedua. Gambar ketiga menunjukkan pemilihan tanggal menggunakan kalender, memastikan pengguna dapat memilih tanggal check-in dengan mudah. Setelah mengisi semua informasi yang diperlukan, seperti ditunjukkan pada gambar keempat dengan contoh pengisian dari Jakarta, pengguna juga dapat memilih jenis kamar yang diinginkan dari beberapa opsi yang tersedia. Terakhir, seperti ditunjukkan pada gambar kelima, pengguna menambahkan jumlah malam menginap dan menekan tombol “Search” untuk memfilter hotel sesuai dengan preferensi yang telah mereka masukkan. Fitur ini dirancang untuk memberikan pengalaman pemesanan hotel

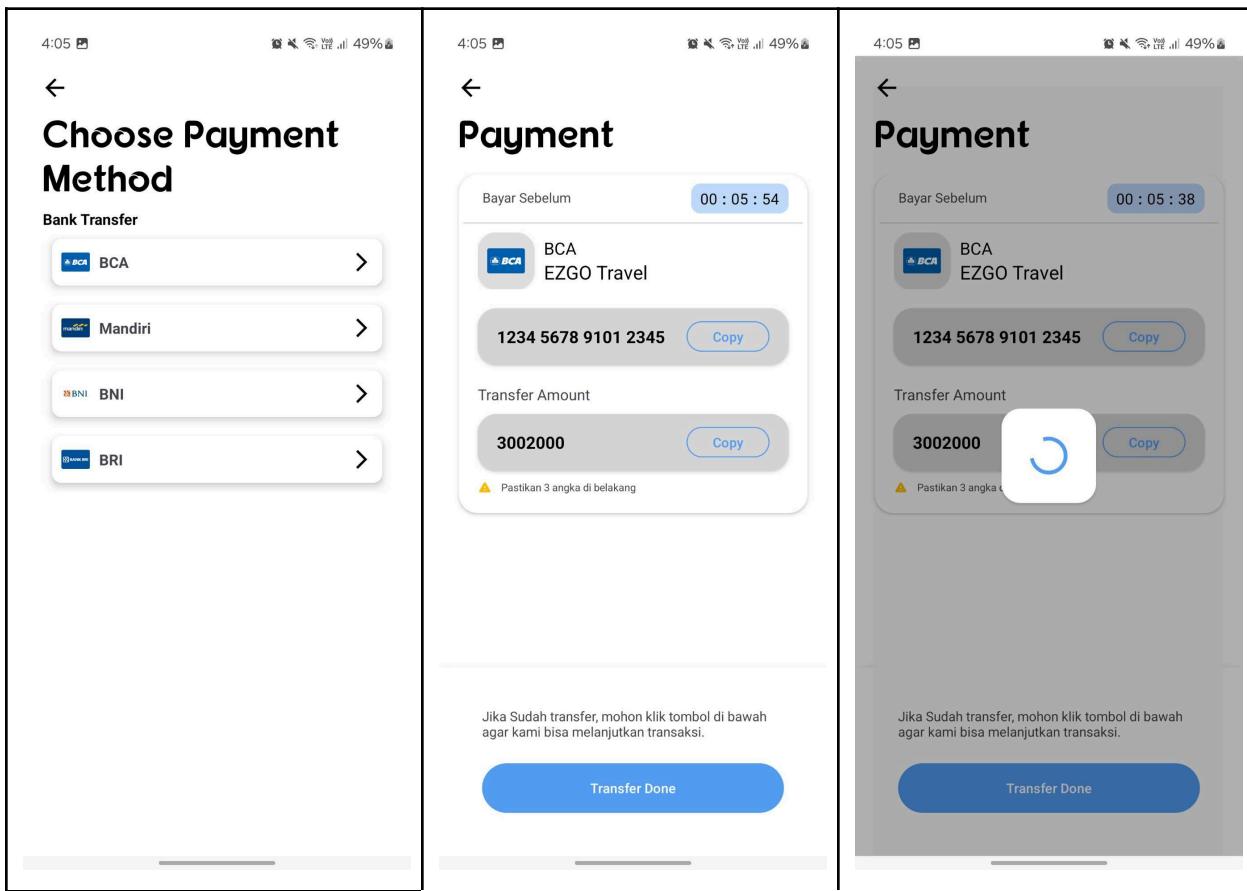
yang cepat dan optimal, memastikan bahwa pengguna dapat menemukan dan memesan kamar hotel dengan mudah sesuai kebutuhan perjalanan mereka.

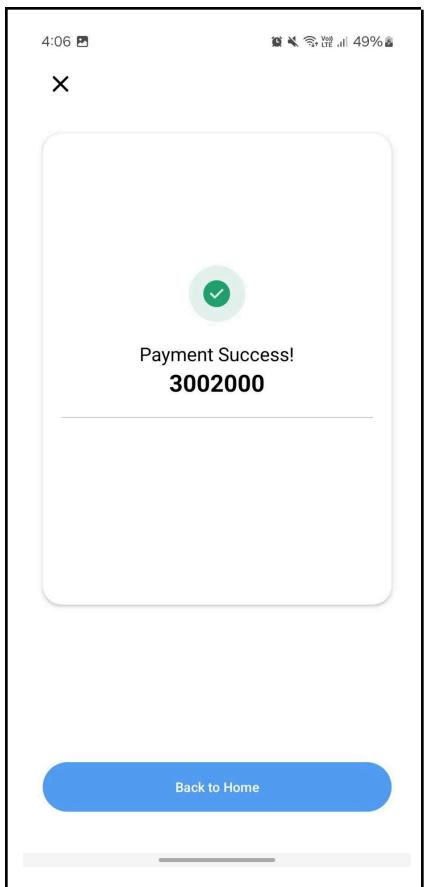
- **Ketika Pengguna Sudah Memilih Hotel**



Tampilan dua gambar UI di atas adalah tampilan *Main Menu (Fitur Pembelian Hotel)* EZGO khususnya ketika aplikasi menunjukkan beberapa pilihan hotel yang sesuai dengan input pengguna, lalu pengguna dapat memilih hotel tersebut. Setelah pengguna memasukkan informasi pencarian hotel dan menekan tombol “Search”, aplikasi akan *menampilkan hasil pencarian hotel yang sesuai*, seperti yang terlihat pada gambar pertama dengan pilihan “Luxury Hotel Jakarta” beserta detail tanggal check-in, jumlah malam menginap, jenis kamar, dan harga per malam. Pengguna kemudian dapat memilih hotel yang diinginkan dan melihat halaman detail pembelian yang menampilkan rincian harga per malam, jumlah malam menginap, biaya tambahan, dan total harga. Pengguna memiliki *opsi untuk menambah malam menginap* atau langsung memesan kamar dengan menekan tombol “Order Now”.

- Detail Proses Pembayaran

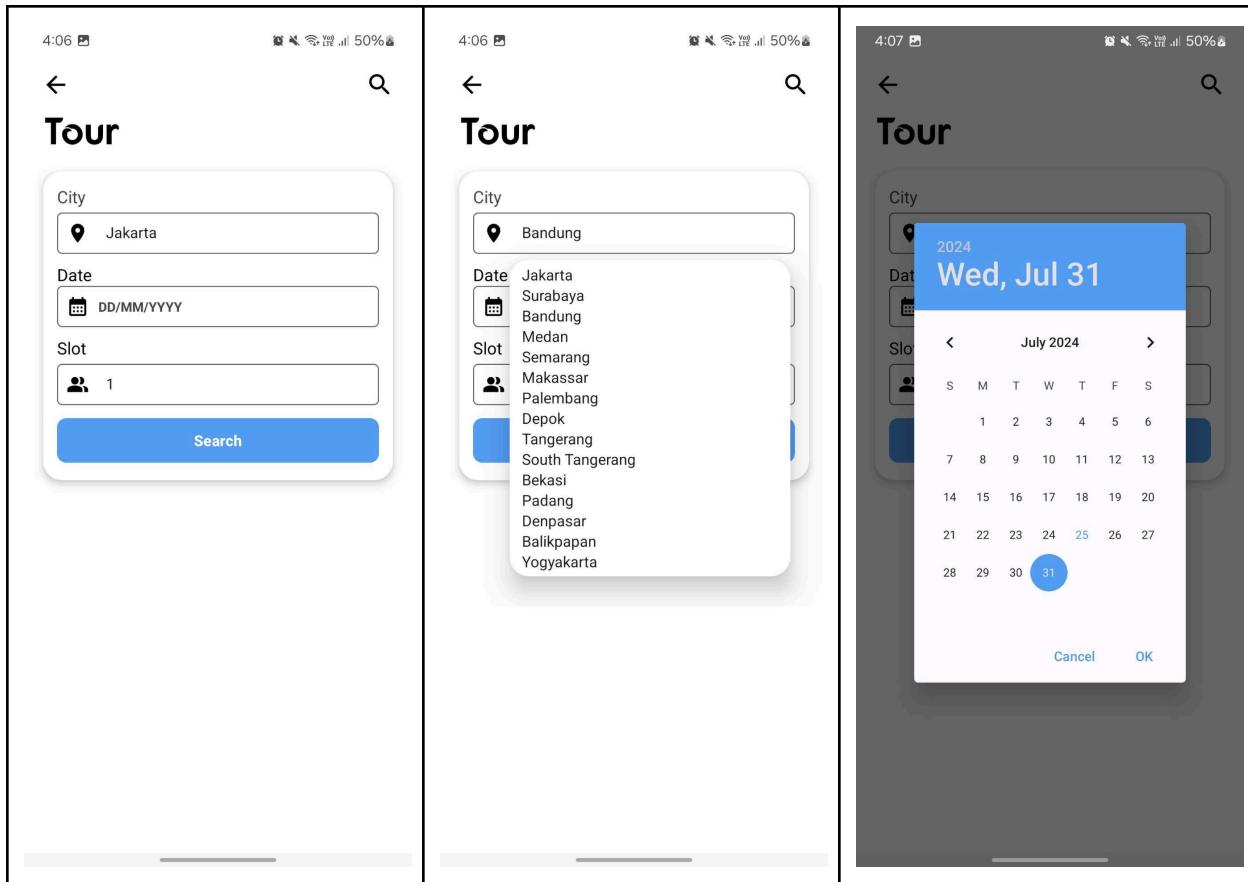


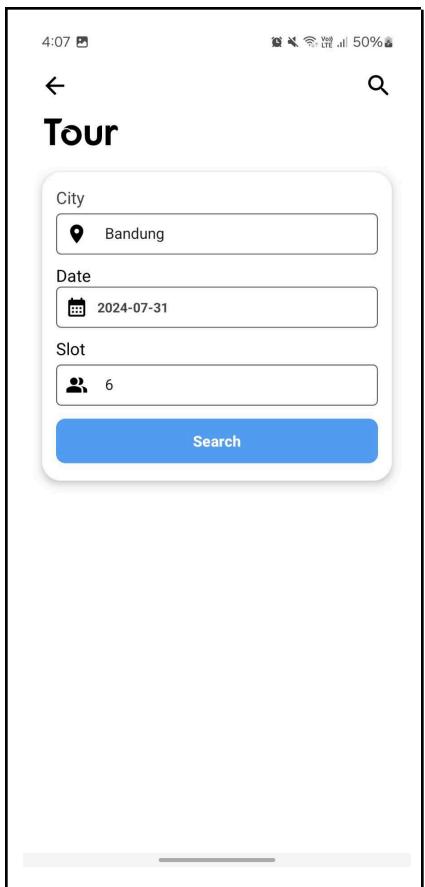


Tampilan empat gambar UI di atas adalah tampilan *Main Menu (Fitur Pembelian Hotel)* EZGO khususnya pada detail proses pembayaran. Setelah pengguna memilih hotel dan melanjutkan ke pembayaran, aplikasi akan menyediakan *berbagai metode pembayaran* untuk memudahkan transaksi. Gambar pertama menampilkan pilihan bank untuk transfer, seperti BCA, Mandiri, BNI, dan BRI. Pengguna dapat memilih salah satu bank, kemudian diarahkan ke halaman detail pembayaran yang menampilkan informasi *nomor rekening bank tujuan* dan *jumlah transfer*, lengkap dengan tombol “*Copy*” untuk memudahkan penyalinan informasi. Pada gambar kedua, pengguna dapat melihat *informasi transfer dengan batas waktu pembayaran* yang tercantum. Setelah melakukan transfer, pengguna menekan tombol “*Transfer Done*” untuk mengkonfirmasi pembayaran. Gambar ketiga menunjukkan proses verifikasi, dan seperti pada gambar keempat, pengguna akan melihat notifikasi “*Payment Success!*” yang menandakan bahwa pembayaran telah berhasil. Fitur ini memastikan bahwa pengguna dapat melakukan transaksi dengan aman dan mudah melalui berbagai pilihan metode pembayaran yang disediakan oleh EZGO.

### **3.1.7. Main Menu (Fitur Pembelian Paket Tour)**

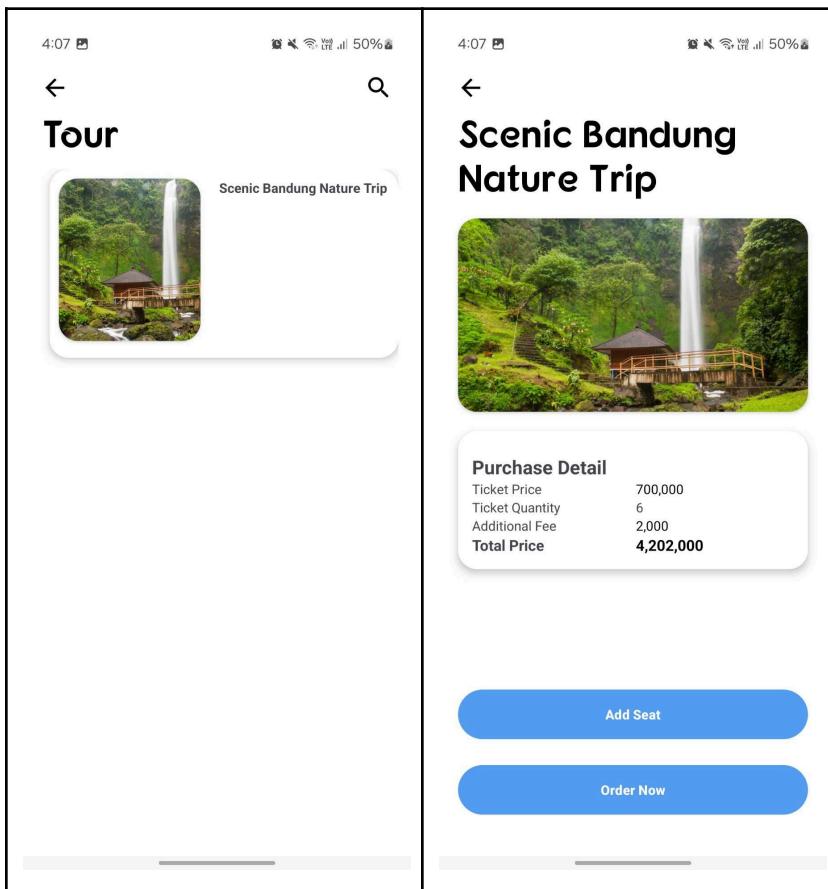
- **Pengguna Mengisi Form Pencarian dan Pembelian Paket Tour**





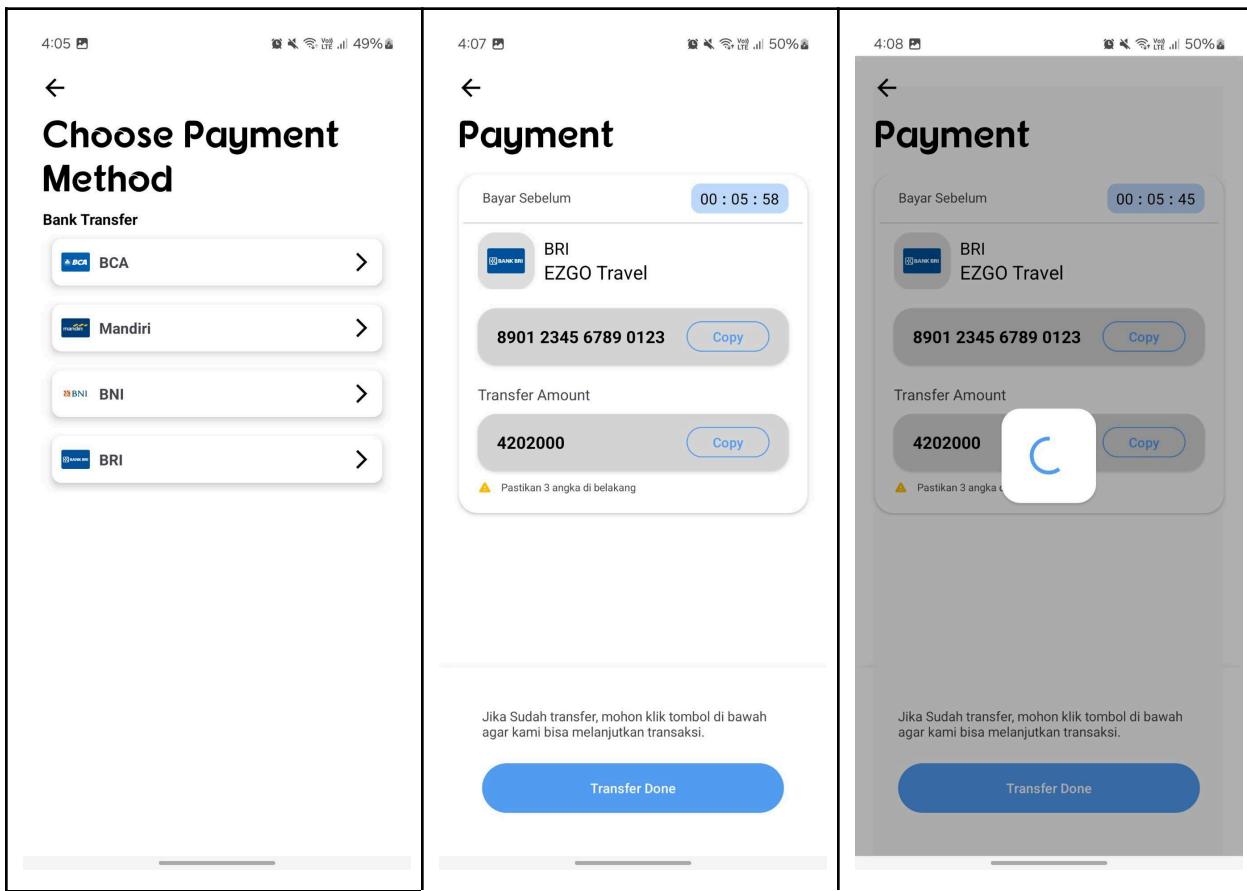
Pada tampilan UI *Main Menu (Fitur Pembelian Paket Tour)* EZGO khususnya pada form pencarian dan pembelian paket tour seperti pada empat gambar di atas, pengguna dapat dengan mudah melakukan pencarian dan pemesanan paket tour sesuai kebutuhan mereka. Pada gambar pertama, pengguna diminta untuk *mengisi informasi* seperti kota tujuan (“*City*”), tanggal tour (“*Date*”), dan jumlah peserta yang mengikuti tour (“*Slot*”). Pengguna dapat memilih dari berbagai kota tujuan yang tersedia, seperti yang ditampilkan pada gambar kedua. Gambar ketiga menunjukkan pemilihan tanggal menggunakan kalender, memastikan pengguna dapat memilih tanggal tour dengan mudah. Setelah mengisi semua informasi yang diperlukan, seperti ditunjukkan pada gambar keempat dengan contoh pengisian kota Bandung, tanggal 31 Juli 2024, dan jumlah peserta 6 orang, pengguna dapat menekan tombol “*Search*” untuk memfilter paket tour yang tersedia sesuai dengan preferensi yang telah mereka masukkan. Fitur ini dirancang untuk memberikan pengalaman pemesanan paket tour yang cepat dan optimal, memastikan bahwa pengguna dapat menemukan dan memesan paket tour dengan mudah sesuai kebutuhan perjalanan mereka.

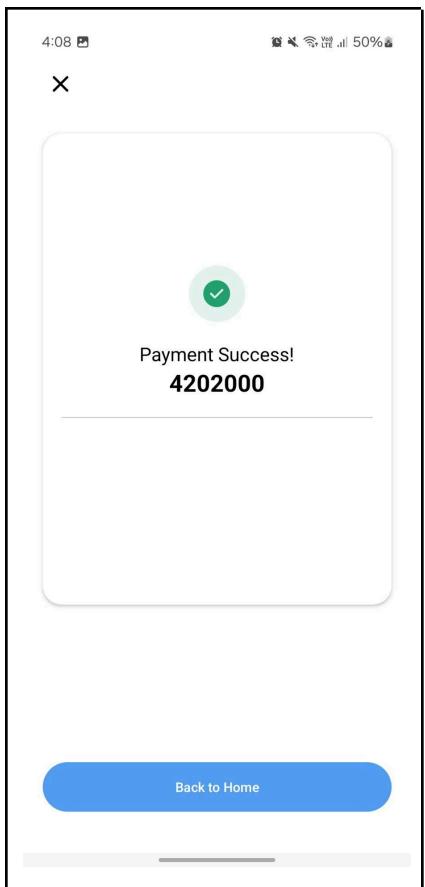
- **Ketika Pengguna Sudah Memilih Paket Tour**



Tampilan dua gambar UI di atas adalah tampilan *Main Menu (Fitur Pembelian Paket Tour)* EZGO khususnya ketika aplikasi menunjukkan beberapa pilihan paket tour yang sesuai dengan input pengguna, lalu pengguna dapat memilih paket tour tersebut. Setelah pengguna memasukkan informasi pencarian tour dan menekan tombol “Search”, aplikasi akan menampilkan hasil pencarian tour yang sesuai, seperti yang terlihat pada gambar pertama dengan pilihan “Scenic Bandung Nature Trip” beserta detail tanggal, jumlah peserta, dan harga per orang. Pengguna kemudian dapat memilih paket tour yang diinginkan dan melihat halaman detail pembelian yang menampilkan rincian harga per orang, jumlah peserta, biaya tambahan, dan total harga. Pengguna memiliki opsi untuk menambah jumlah peserta atau langsung memesan tour dengan menekan tombol “Order Now”, memastikan proses pemesanan paket tour yang intuitif dan transparan.

- Detail Proses Pembayaran

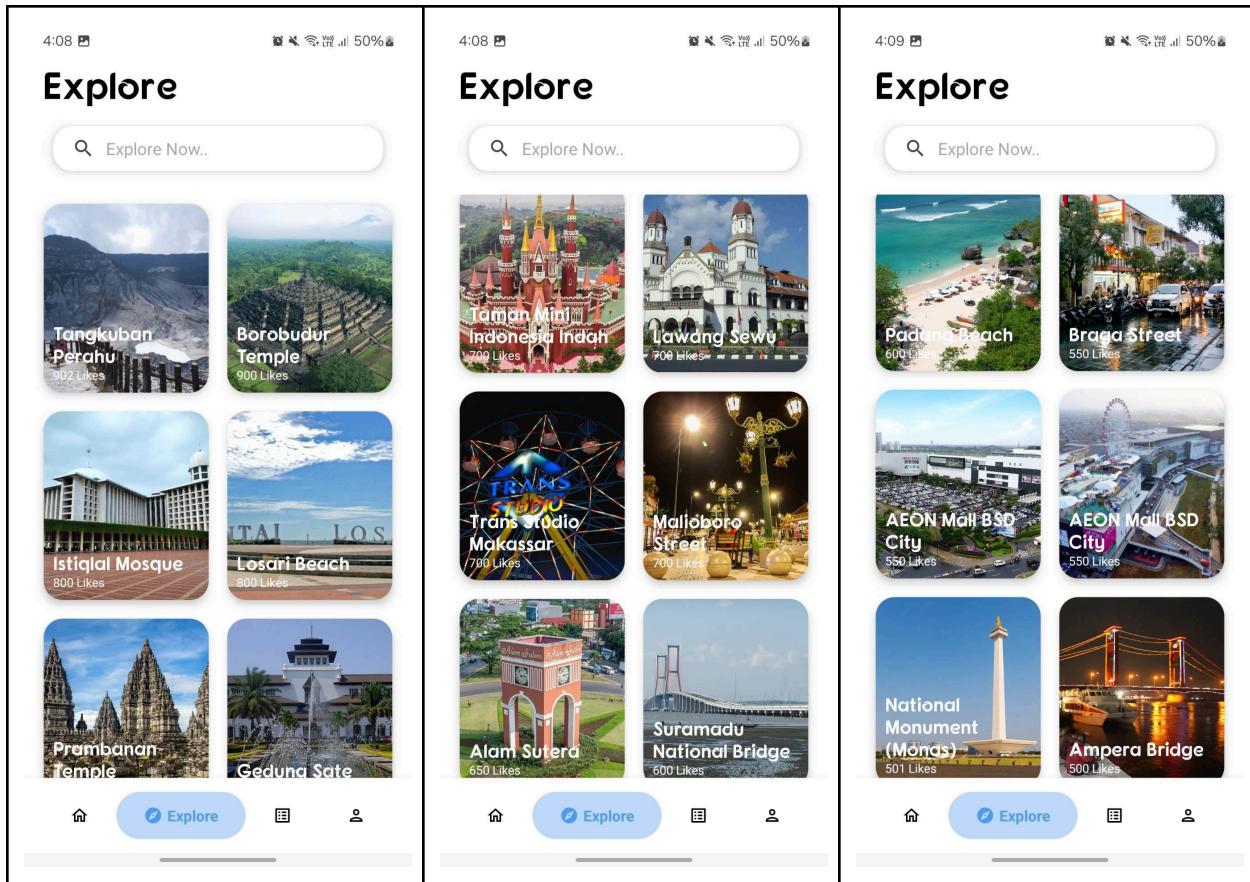


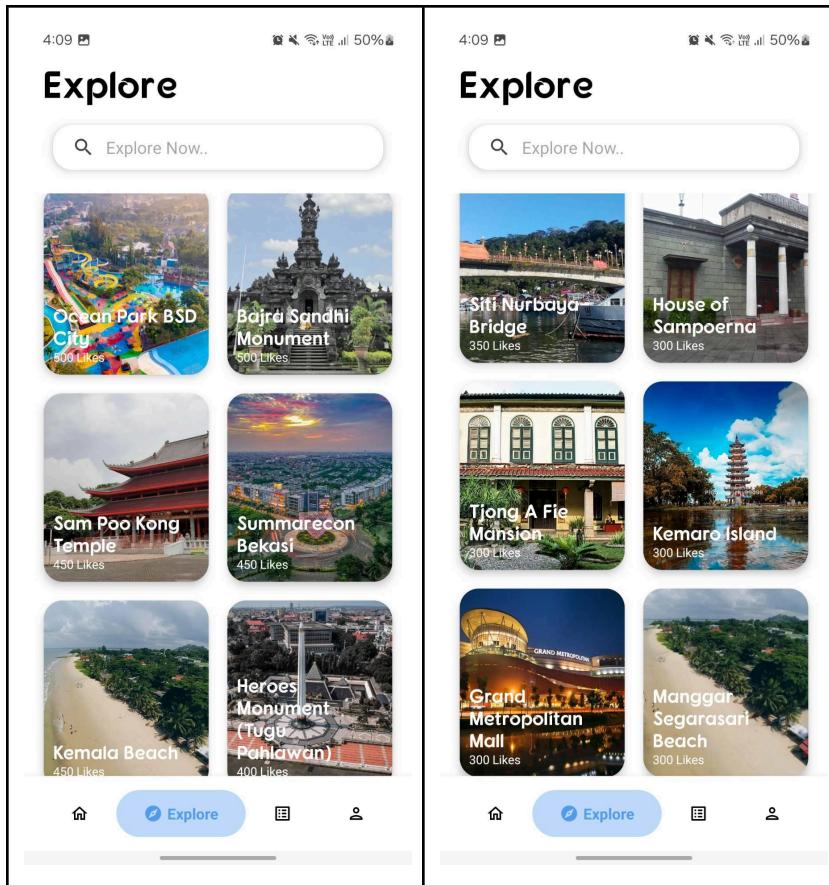


Tampilan empat gambar UI di atas adalah tampilan *Main Menu (Fitur Pembelian Paket Tour)* EZGO khususnya pada detail proses pembayaran. Setelah pengguna memilih paket tour dan melanjutkan ke pembayaran, aplikasi akan menyediakan *berbagai metode pembayaran* untuk memudahkan transaksi. Gambar pertama menampilkan pilihan bank untuk transfer, seperti BCA, Mandiri, BNI, dan BRI. Pengguna dapat memilih salah satu bank, kemudian diarahkan ke halaman detail pembayaran yang menampilkan informasi *nomor rekening bank tujuan* dan *jumlah transfer*, lengkap dengan tombol “*Copy*” untuk memudahkan penyalinan informasi. Pada gambar kedua, pengguna dapat melihat *informasi transfer dengan batas waktu pembayaran* yang tercantum. Setelah melakukan transfer, pengguna menekan tombol “*Transfer Done*” untuk mengkonfirmasi pembayaran. Gambar ketiga menunjukkan proses verifikasi, dan seperti pada gambar keempat, pengguna akan melihat notifikasi “*Payment Success!*” yang menandakan bahwa pembayaran telah berhasil. Fitur ini memastikan bahwa pengguna dapat melakukan transaksi dengan aman dan mudah melalui berbagai pilihan metode pembayaran yang disediakan oleh EZGO.

### **3.1.8. Menu Explore Destinasi Wisata**

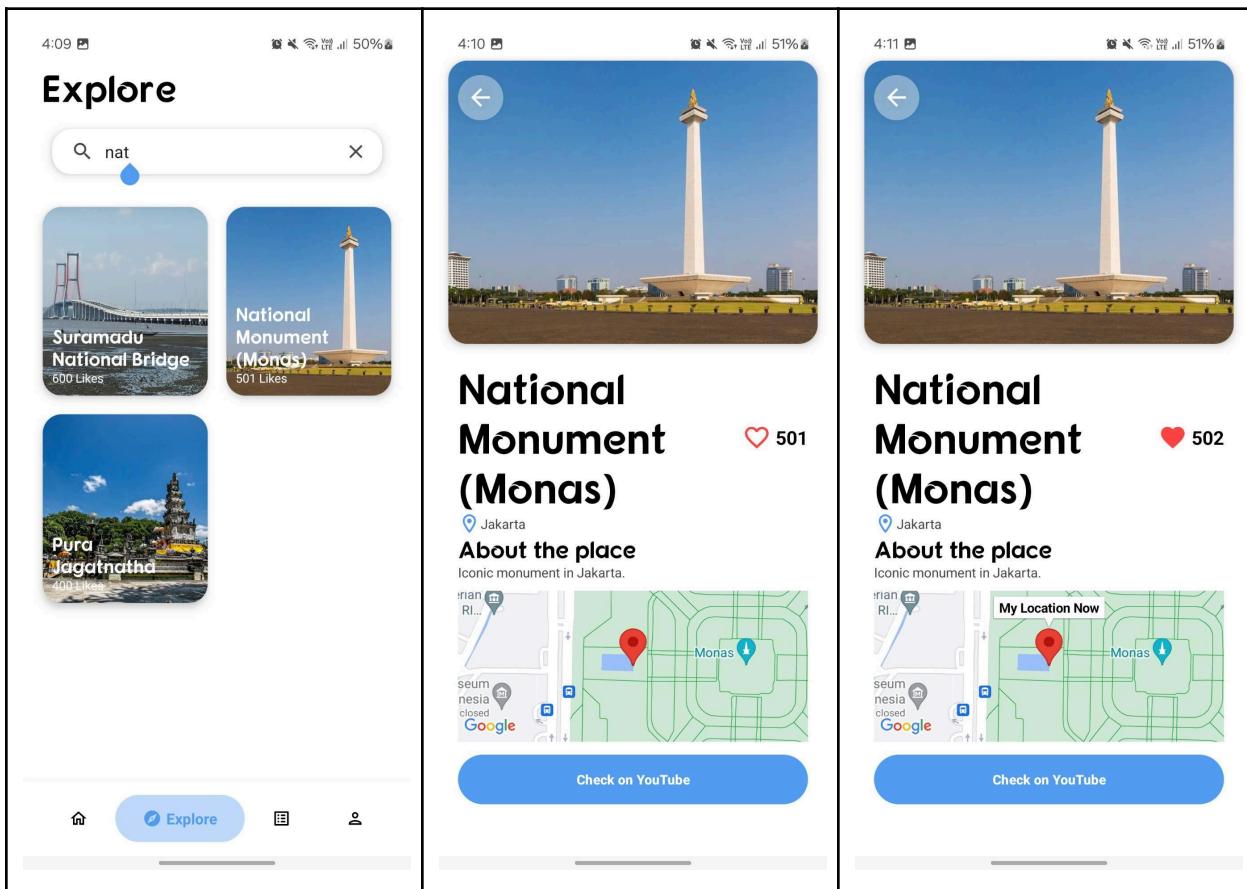
- **Tampilan Menu Explore**

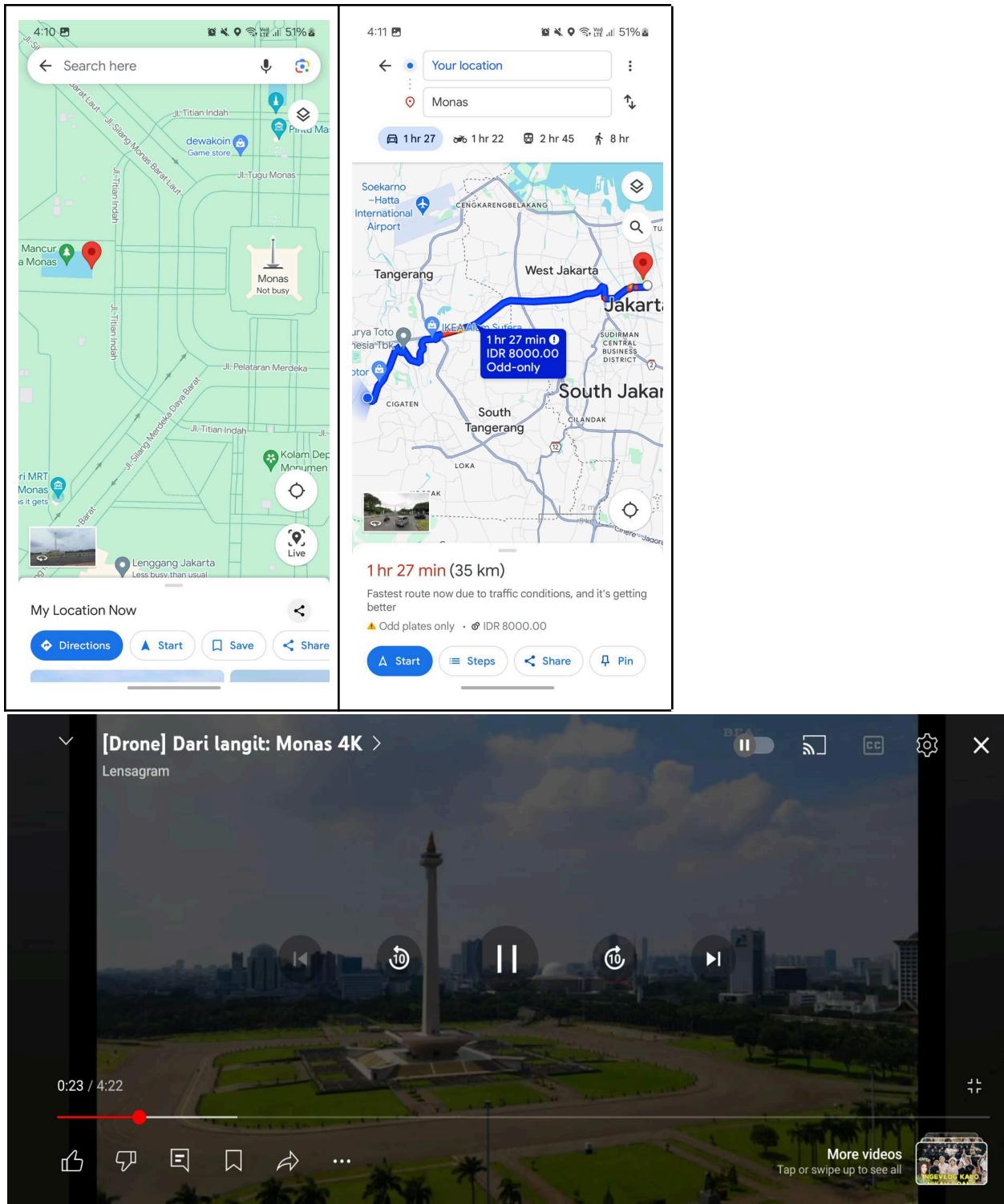




Pada tampilan UI *Menu Explore* aplikasi EZGO, pengguna dapat melakukan eksplorasi terhadap berbagai destinasi wisata yang beragam dengan cara *scroll*. Pada gambar pertama hingga kelima, terlihat berbagai destinasi populer di Indonesia yang disusun dalam grid. Destinasi-destinasi ini mencakup tempat-tempat seperti Tangkuban Perahu, Candi Borobudur, Masjid Istiqlal, Pantai Losari, Taman Mini Indonesia Indah, Lawang Sewu, Monumen Nasional (Monas), dan masih banyak lagi. Setiap destinasi dilengkapi dengan jumlah likes yang memberikan indikasi popularitasnya di antara pengguna. Fitur pencarian “*Explore Now*” di bagian atas memungkinkan pengguna untuk mencari destinasi tertentu secara cepat. Pengguna dapat dengan mudah menggulir ke bawah untuk melihat lebih banyak pilihan destinasi, memastikan bahwa mereka dapat menemukan tempat-tempat menarik untuk dikunjungi sesuai minat mereka. Fitur ini dirancang untuk memberikan pengalaman eksplorasi yang menyenangkan dan informatif, memudahkan pengguna dalam menemukan dan memilih destinasi wisata yang ingin mereka kunjungi.

- Detail Fitur Menu Explore

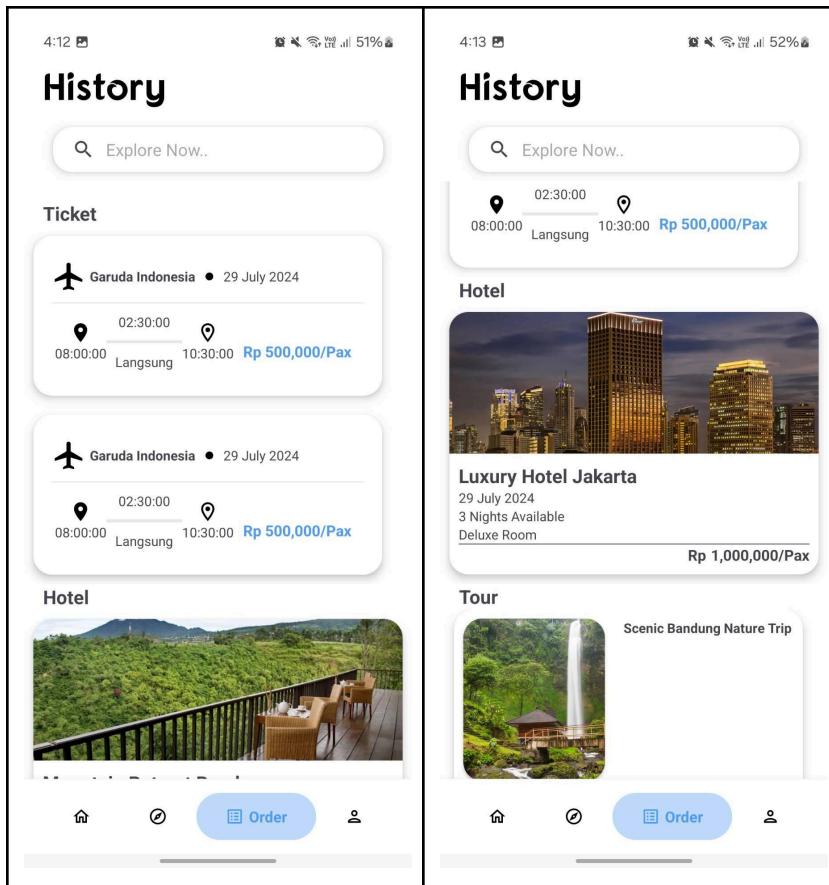




Pada tampilan UI *Detail Fitur Menu Explore Destinasi Wisata EZGO* yang ditunjukkan pada enam gambar di atas, pengguna dapat dengan mudah mencari dan menjelajahi berbagai destinasi wisata. Fitur pencarian ini juga mendukung *pencarian spesifik* dengan menggunakan

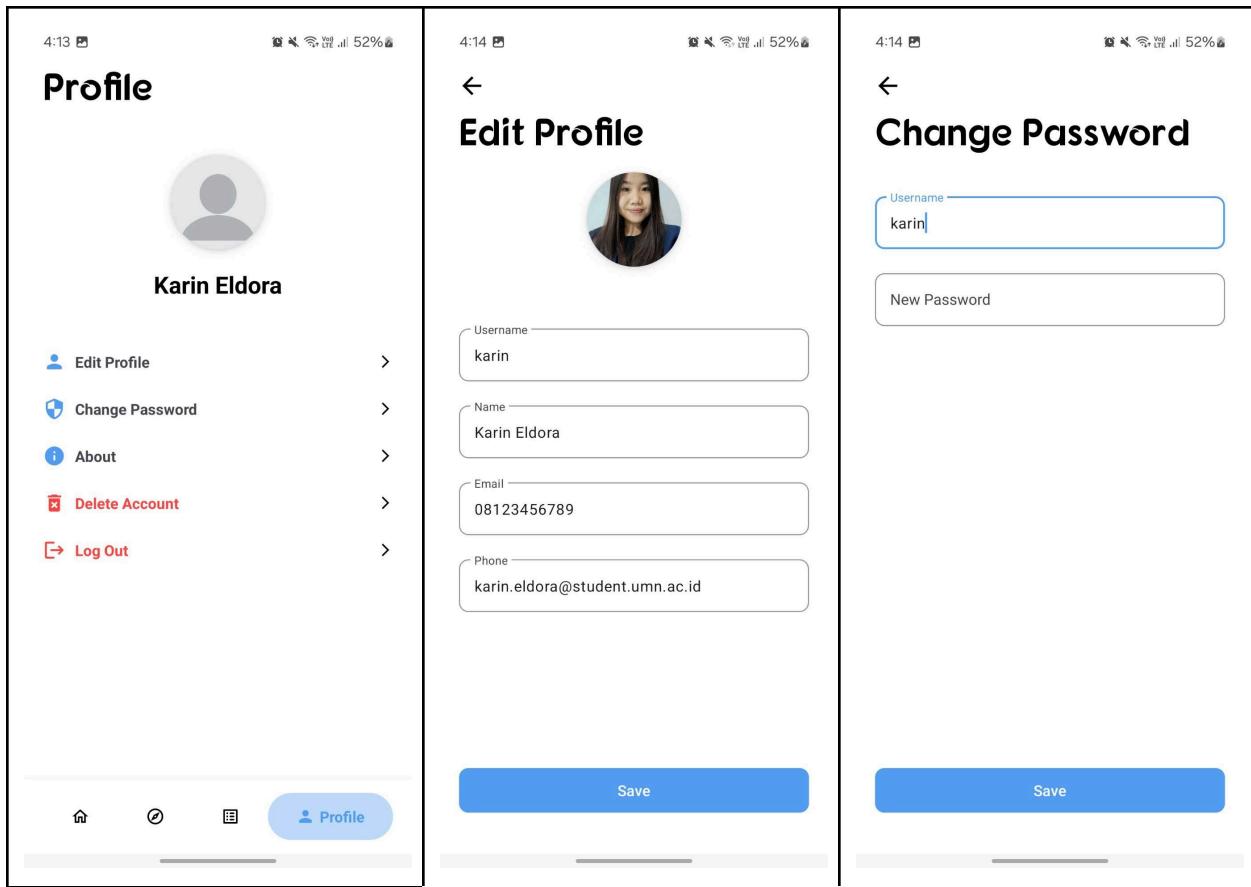
*Search Bar*, sehingga ketika pengguna mulai mengetik kata kunci seperti “nat”, maka hanya produk atau destinasi yang mengandung kata tersebut yang akan ditampilkan, seperti National Monument (Monas), Suramadu National Bridge, dan Pura Jagatnatha. Ketika pengguna mengklik salah satu destinasi, detail destinasi tersebut akan ditampilkan, seperti yang terlihat pada gambar kedua dengan *Monumen Nasional (Monas)*. Detail ini mencakup *informasi mengenai tempat tersebut, foto tempat wisata, lokasi yang ditampilkan di Google Maps, opsi untuk memberikan like*, serta dapat *melihat cuplikan video destinasi tersebut di YouTube*. Gambar ketiga menunjukkan peningkatan jumlah likes setelah pengguna memberikan like pada destinasi. Gambar keempat dan kelima menampilkan *peta* dari *Google Maps* yang menunjukkan lokasi destinasi dan rute perjalanan dari lokasi pengguna saat ini. Terakhir, gambar keenam menampilkan *video destinasi di YouTube*, memberikan pengguna visualisasi yang lebih mendalam tentang tempat yang mereka ingin kunjungi. Fitur ini dirancang untuk mempermudah pengguna dalam menemukan, mengeksplorasi, dan merencanakan kunjungan ke berbagai destinasi wisata dengan cepat dan informatif.

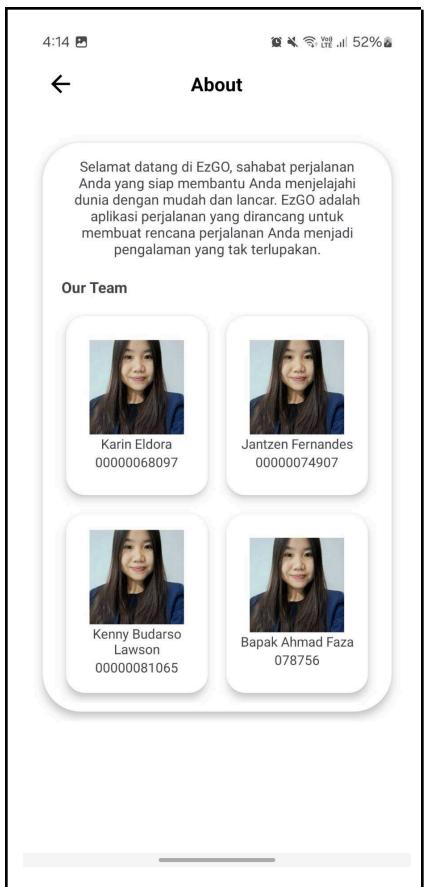
### **3.1.9. Menu Lihat History Pembelian (Order)**



Pada tampilan UI *Menu Lihat History Pembelian (Order)* di EZGO yang ditunjukkan pada dua gambar di atas, pengguna dapat melihat riwayat pembelian mereka yang dikelompokkan dalam tiga kategori utama: *Ticket*, *Hotel*, dan *Tour*. Pada gambar pertama, terlihat bahwa pembelian *tiket* pesawat ditampilkan dalam grid pertama dengan rincian penerbangan seperti maskapai, tanggal, waktu keberangkatan dan tiba, serta harga per penumpang. Gambar kedua menunjukkan riwayat pembelian *hotel* yang ditampilkan dalam grid kedua dengan detail nama hotel, tanggal check-in, jumlah malam menginap, jenis kamar, dan harga per malam. Selain itu, riwayat pembelian *paket tour* juga ditampilkan dalam grid ketiga dengan informasi tentang nama paket tour. Setiap kategori memudahkan pengguna untuk melihat dan mengelola history pembelian mereka dengan jelas dan terorganisir. Fitur ini dirancang untuk memberikan pengalaman pengguna yang nyaman dalam mengakses dan meninjau pembelian yang telah mereka lakukan melalui aplikasi EZGO.

### **3.1.10. Menu Profile**





Pada tampilan UI *Menu Profile* untuk pengguna EZGO yang ditunjukkan pada empat gambar di atas, pengguna dapat mengakses dan mengelola berbagai informasi pribadi serta melakukan beberapa aktivitas penting. Gambar pertama menunjukkan tampilan utama profil yang menampilkan *nama pengguna*, opsi untuk *mengedit profil*, *mengubah kata sandi*, *melihat informasi tentang aplikasi (About)*, *menghapus akun*, dan *keluar (log out)*. Gambar kedua menunjukkan layar *Edit Profile* di mana pengguna dapat mengubah informasi seperti *nama pengguna*, *nama lengkap*, *email*, dan *nomor telepon*. Gambar ketiga menampilkan layar *Change Password* yang memungkinkan pengguna untuk *mengubah kata sandi* mereka dengan *memasukkan kata sandi baru*. Gambar keempat memperlihatkan layar *About* yang memberikan informasi tentang aplikasi EZGO dan menampilkan anggota tim yang mengembangkan aplikasi ini. Fitur *Delete Account*, yang ditampilkan dalam warna merah menandakan tindakan yang serius, memungkinkan pengguna untuk menghapus akun mereka secara permanen dari aplikasi. Jika pengguna memilih untuk menghapus akun, semua data pengguna terkait akan dihapus dari database, termasuk informasi pribadi, riwayat pembelian, dan preferensi pengguna, sehingga

tidak ada jejak yang tersisa dalam sistem. Di sisi lain, fitur *Log Out* memberikan opsi bagi pengguna untuk keluar dari sesi aplikasi mereka tanpa menghapus data. Ketika pengguna memilih untuk log out, mereka hanya akan keluar dari akun mereka di perangkat tersebut, tetapi semua data dan informasi pribadi tetap tersimpan dalam database. Fitur-fitur ini dirancang untuk memberikan pengguna kontrol penuh atas informasi pribadi mereka dan memudahkan mereka dalam mengelola akun mereka dengan aman.

## 3.2. Code Aplikasi Android (Android Studio)

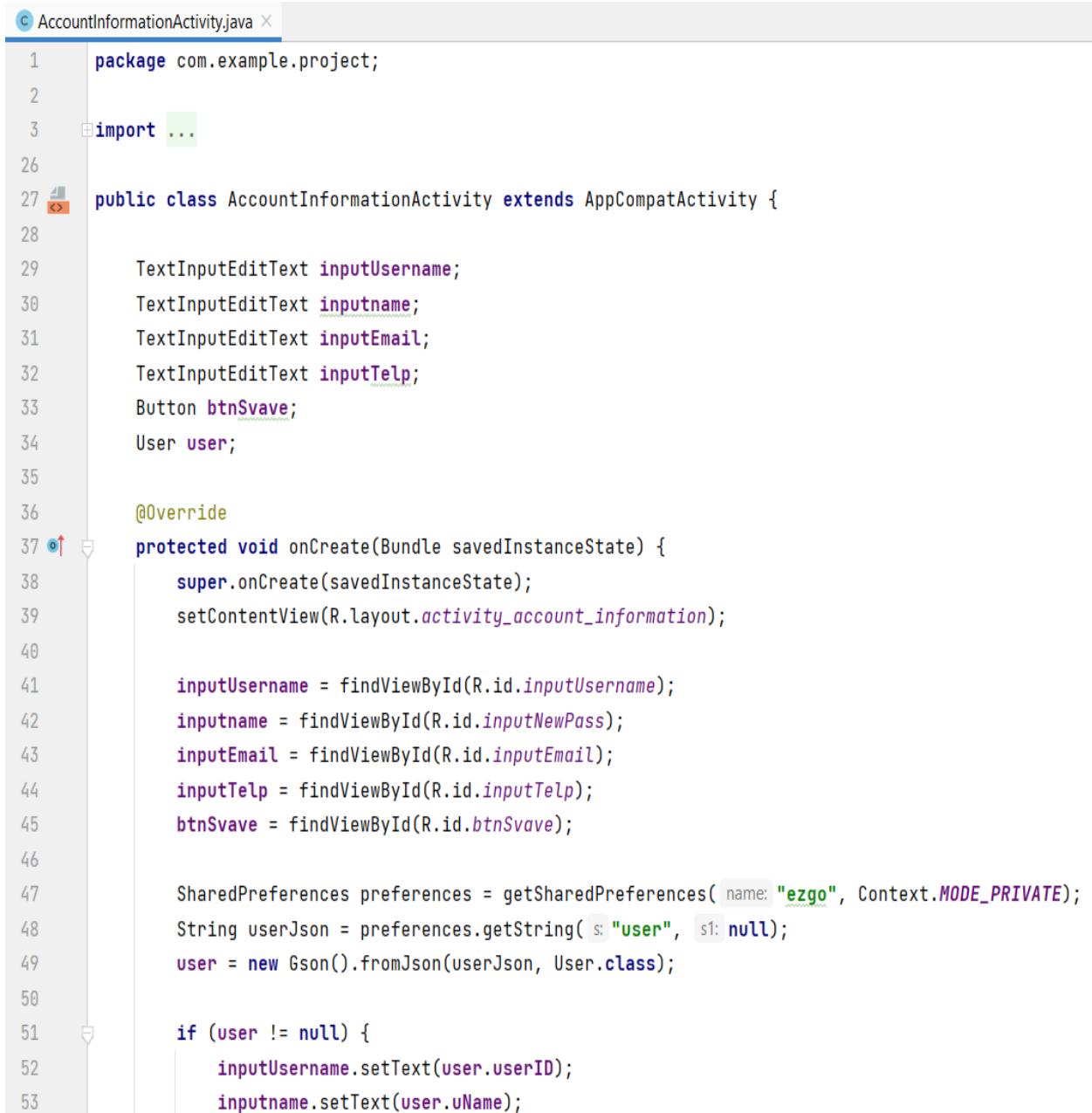
### 3.2.1. AboutActivity



```
1 package com.example.project;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.FrameLayout;
8
9 public class AboutActivity extends AppCompatActivity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_about);
15         FrameLayout btnBack = findViewById(R.id.backAbout);
16         btnBack.setOnClickListener(view -> onBackPressed());
17     }
18 }
```

Kode java AboutActivity di atas merupakan bagian dari kelas *AboutActivity* dalam aplikasi Android yang mengelola aktivitas “Tentang”. Kelas ini memperluas *AppCompatActivity* dan menampilkan tata letak *activity\_about* ketika aktivitas dibuat. Pada metode *onCreate()*, tombol kembali (*btnBack*) yang diwakili oleh *FrameLayout* diidentifikasi dengan *findViewById* menggunakan ID *backAbout*. Kemudian, *OnClickListener* ditetapkan pada tombol tersebut untuk memanggil metode *onBackPressed()* saat tombol diklik, yang akan mengembalikan pengguna ke aktivitas sebelumnya. Kode ini mengimplementasikan fitur untuk menampilkan halaman tentang aplikasi dan menyediakan navigasi kembali ke halaman sebelumnya saat tombol kembali ditekan.

### 3.2.2. AccountInformationActivity



```
1 package com.example.project;
2
3 import ...
4
5
6 public class AccountInformationActivity extends AppCompatActivity {
7
8
9     TextInputEditText inputUsername;
10    TextInputEditText inputname;
11    TextInputEditText inputEmail;
12    TextInputEditText inputTelp;
13    Button btnSvave;
14    User user;
15
16
17    @Override
18    protected void onCreate(Bundle savedInstanceState) {
19        super.onCreate(savedInstanceState);
20        setContentView(R.layout.activity_account_information);
21
22        inputUsername = findViewById(R.id.inputUsername);
23        inputname = findViewById(R.id.inputNewPass);
24       inputEmail = findViewById(R.id.inputEmail);
25        inputTelp = findViewById(R.id.inputTelp);
26        btnSvave = findViewById(R.id.btnSvave);
27
28
29        SharedPreferences preferences = getSharedPreferences("ezgo", Context.MODE_PRIVATE);
30        String userJson = preferences.getString("user", null);
31        user = new Gson().fromJson(userJson, User.class);
32
33        if (user != null) {
34            inputUsername.setText(user.userID);
35            inputname.setText(user.uName);
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
980
981
982
983
984
985
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1167
1168
1169
1170
1171
1172
1173
1174
1175
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1208
1209
1210
1211
1212
1213
1214
1215
1216
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1266
1267
1268
1268
1269
1270
1271
1272
1273
1274
1275
1275
1276
1277
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1294
1295
1296
1296
1297
1298
1298
1299
1300
1301
1302
1303
1304
1305
1305
1306
1307
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1315
1316
1317
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1325
1326
1327
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1335
1336
1337
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1345
1346
1347
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1355
1356
1357
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1365
1366
1367
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1375
1376
1377
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1394
1395
1396
1396
1397
1398
1398
1399
1400
1401
1402
1403
1404
1404
1405
1406
1406
1407
1408
1408
1409
1410
1410
1411
1412
1412
1413
1414
1414
1415
1416
1416
1417
1418
1418
1419
1420
1420
1421
1422
1422
1423
1424
1424
1425
1426
1426
1427
1428
1428
1429
1430
1430
1431
1432
1432
1433
1434
1434
1435
1436
1436
1437
1438
1438
1439
1440
1440
1441
1442
1442
1443
1444
1444
1445
1446
1446
1447
1448
1448
1449
1450
1450
1451
1452
1452
1453
1454
1454
1455
1456
1456
1457
1458
1458
1459
1460
1460
1461
1462
1462
1463
1464
1464
1465
1466
1466
1467
1468
1468
1469
1470
1470
1471
1472
1472
1473
1474
1474
1475
1476
1476
1477
1478
1478
1479
1480
1480
1481
1482
1482
1483
1484
1484
1485
1486
1486
1487
1488
1488
1489
1490
1490
1491
1492
1492
1493
1494
1494
1495
1496
1496
1497
1498
1498
1499
1500
1500
1501
1502
1502
1503
1504
1504
1505
1506
1506
1507
1508
1508
1509
1510
1510
1511
1512
1512
1513
1514
1514
1515
1516
1516
1517
1518
1518
1519
1520
1520
1521
1522
1522
1523
1524
1524
1525
1526
1526
1527
1528
1528
1529
1530
1530
1531
1532
1532
1533
1534
1534
1535
1536
1536
1537
1538
1538
1539
1540
1540
1541
1542
1542
1543
1544
1544
1545
1546
1546
1547
1548
1548
1549
1550
1550
1551
1552
1552
1553
1554
1554
1555
1556
1556
1557
1558
1558
1559
1560
1560
1561
1562
1562
1563
1564
1564
1565
1566
1566
1567
1568
1568
1569
1570
1570
1571
1572
1572
1573
1574
1574
1575
1576
1576
1577
1578
1578
1579
1580
1580
1581
1582
1582
1583
1584
1584
1585
1586
1586
1587
1588
1588
1589
1590
1590
1591
1592
1592
1593
1594
1594
1595
1596
1596
1597
1598
1598
1599
1600
1600
1601
1602
1602
1603
1604
1604
1605
1606
1606
1607
1608
1608
1609
1610
1610
1611
1612
1612
1613
1614
1614
1615
1616
1616
1617
1618
1618
1619
1620
1620
1621
1622
1622
1623
1624
1624
1625
1626
1626
1627
1628
1628
1629
1630
1630
1631
1632
1632
1633
1634
1634
1635
1636
1636
1637
1638
1638
1639
1640
1640
1641
1642
1642
1643
1644
1644
1645
1646
1646
1647
1648
1648
1649
1650
1650
1651
1652
1652
1653
1654
1654
1655
1656
1656
1657
1658
1658
1659
1660
1660
1661
1662
1662
1663
1664
1664
1665
1666
1666
1667
1668
1668
1669
1670
1670
1671
1672
1672
1673
1674
1674
1675
1676
1676
1677
1678
1678
1679
1680
1680
1681
1682
1682
1683
1684
1684
1685
1686
1686
1687
1688
1688
1689
1690
1690
1691
1692
1692
1693
1694
1694
1695
1696
1696
1697
1698
1698
1699
1700
1700
1701
1702
1702
1703
1704
1704
1705
1706
1706
1707
1708
1708
1709
1710
1710
1711
1712
1712
1713
1714
1714
1715
1716
1716
1717
1718
1718
1719
1720
1720
1721
1722
1722
1723
1724
1724
1725
1726
1726
1727
1728
1728
1729
1730
1730
1731
1732
1732
1733
1734
1734
1735
1736
1736
1737
1738
1738
1739
1740
1740
1741
1742
1742
1743
1744
1744
1745
1746
1746
1747
1748
1748
1749
1750
1750
1751
1752
1752
1753
1754
1754
1755
1756
1756
1757
1758
1758
1759
1760
1760
1761
1762
1762
1763
1764
1764
1765
1766
1766
1767
1768
1768
1769
1770
1770
1771
1772
1772
1773
1774
1774
1775
1776
1776
1777
1778
1778
1779
1780
1780
1781
1782
1782
1783
1784
1784
1785
1786
1786
1787
1788
1788
1789
1790
1790
1791
1792
1792
1793
1794
1794
1795
1796
1796
1797
1798
1798
1799
1800
1800
1801
1802
1802
1803
1804
1804
1805
1806
1806
1807
1808
1808
1809
1810
1810
1811
1812
1812
1813
1814
1814
1815
1816
1816
1817
1818
1818
1819
1820
1820
1821
1822
1822
1823
1824
1824
1825
1826
1826
1827
1828
1828
1829
1830
1830
1831
1832
1832
1833
1834
1834
1835
1836
1836
1837
1838
1838
1839
1840
1840
1841
1842
1842
1843
1844
1844
1845
1846
1846
1847
1848
1848
1849
1850
1850
1851
1852
1852
1853
1854
1854
1855
1856
1856
1857
1858
1858
1859
1860
1860
1861
1862
1862
1863
1864
1864
1865
1866
1866
1867
1868
1868
1869
1870
1870
1871
1872
1872
1873
1874
1874
1875
1876
1876
1877
1878
1878
1879
1880
1880
1881
1882
1882
1883
1884
1884
1885
1886
1886
1887
1888
1888
1889
1890
1890
1891
1892
1892
1893
1894
1894
1895
1896
1896
1897
1898
1898
1899
1900
1900
1901
1902
1902
1903
1904
1904
1905
1906
1906
1907
1908
1908
1909
1910
1910
1911
1912
1912
1913
1914
1914
1915
1916
1916
1917
1918
1918
1919
1920
1920
1921
1922
1922
1923
1924
1924
1925
1926
1926
1927
1928
1928
1929
1930
1930
1931
1932
1932
1933
1934
1934
1935
1936
1936
1937
1938
1938
1939
1940
1940
1941
1942
1942
1943
1944
1944
1945
1946
1946
1947
1948
1948
1949
1950
1950
1951
1952
1952
1953
1954
1954
1955
1956
1956
1957
1958
1958
1959
1960
1960
1961
1962
1962
1963
1964
1964
1965
1966
1966
1967
1968
1968
1969
1970
1970
1971
1972
1972
1973
1974
1974
1975
1976
1976
1977
1978
1978
1979
1980
1980
1981
1982
1982
1983
1984
1984
1985
1986
1986
1987
1988
1988
1989
1990
1990
1991
1992
1992
1993
1994
1994
1995
1996
1
```

```
© AccountInformationActivity.java ×
54     inputEmail.setText(user.uPhone);
55     inputTelp.setText(user.uEmail);
56
57     btnSave.setOnClickListener(new View.OnClickListener() {
58         @Override
59         public void onClick(View view) {
60             String newUsername = inputUsername.getText().toString();
61             String newName = inputname.getText().toString();
62             String newEmail = inputTelp.getText().toString();
63             String newPhone = inputEmail.getText().toString();
64
65             user.userID = newUsername;
66             user.uName = newName;
67             user.uEmail = newEmail;
68             user.uPhone = newPhone;
69
70             SharedPreferences.Editor editor = preferences.edit();
71             editor.putString("user", new Gson().toJson(user));
72             editor.apply();
73
74             RequestQueue queue = Volley.newRequestQueue(context: AccountInformationActivity.this);
75             Gson gson = new Gson();
76
77             Map<String, Object> params = new HashMap<>();
78             params.put("controller", "account");
79             params.put("method", "update");
80             params.put("userID", user.userID);
81             params.put("uUsername", newUsername);
82             params.put("uName", newName);
83             params.put("uEmail", newEmail);
84             params.put("uTel", newPhone);
```

```
© AccountInformationActivity.java ×
86     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST,
87             new JSONObject(params),
88             new Response.Listener<JSONObject>() {
89                 @Override
90                 public void onResponse(JSONObject response) {
91                     Log.d(tag: "Ezgo", msg: "Response: " + response);
92                     ResponseNoObject resp = gson.fromJson(response.toString(), new TypeToken<ResponseNoObject>() {
93                         .getType());
94                     boolean success = resp.isSuccess();
95
96                     if (success) {
97                         Toast.makeText(getApplicationContext(), text: "Data Updated Successfully", Toast.LENGTH_LONG).show();
98                         finish();
99                     } else {
100                         Toast.makeText(getApplicationContext(), text: "Failed to Update Data", Toast.LENGTH_LONG).show();
101                     }
102                 }
103             },
104             new Response.ErrorListener() {
105                 @Override
106                 public void onErrorResponse(VolleyError error) {
107                     Log.e(tag: "Ezgo", msg: "Error: " + error.toString());
108                     if (error.networkResponse != null) {
109                         String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
110                         Log.e(tag: "Ezgo", msg: "Response: " + responseBody);
111                     }
112                 }
113             });
114             queue.add(jsonObjectRequest);
115         }
116     );
```

Kode java `AccountInformationActivity` di atas merupakan bagian dari kelas `AccountInformationActivity` dalam aplikasi Android yang mengelola tampilan dan pembaruan informasi akun pengguna. Saat aktivitas dibuat, elemen UI seperti `TextInputEditText` dan `Button` diinisialisasi, dan informasi pengguna yang tersimpan dalam `SharedPreferences` dibaca dan ditampilkan. Pengguna dapat mengubah informasi mereka dan menyimpan perubahan tersebut dengan menekan tombol `btnSave`. Saat tombol ditekan, informasi yang diperbarui disimpan kembali ke `SharedPreferences` dan dikirim ke server menggunakan `Volley` dalam bentuk permintaan HTTP POST ke endpoint `router.php`. Jika pembaruan berhasil, aplikasi menampilkan pesan sukses dan menutup aktivitas; jika gagal, menampilkan pesan kesalahan. Kode ini mengimplementasikan fitur untuk melihat dan memperbarui informasi akun pengguna, memungkinkan sinkronisasi data lokal dan server.

### 3.2.3. AdapterHome

```
1 package com.example.project;
2
3 import ...
4
5
6 public class AdapterHome extends RecyclerView.Adapter<AdapterHome.MyViewHolder> {
7     private final Context context;
8     private final ArrayList<MyItem> itemList;
9     private String urlImg = "http://192.168.100.4/ezgo/images/";
10    private Intent intent;
11
12
13    public AdapterHome(Context context, ArrayList<MyItem> itemList) {
14        this.context = context;
15        this.itemList = itemList;
16    }
17
18
19    public static class MyViewHolder extends RecyclerView.ViewHolder {
20        public TextView textTitle;
21        public ImageView cardImg;
22
23        public MyViewHolder(View itemView) {
24            super(itemView);
25            textTitle = itemView.findViewById(R.id.txtCard);
26            cardImg = itemView.findViewById(R.id.ImgDisHome);
27        }
28
29    }
30
31
32    @Override
33    public AdapterHome.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
34        View view = LayoutInflater.from(context).inflate(R.layout.card_home, parent, attachToRoot: false);
35        return new MyViewHolder(view);
36    }
37
38    @Override
39    public void onBindViewHolder(AdapterHome.MyViewHolder holder, int position) {
40        holder.textTitle.setText(itemList.get(position).title);
41        holder.cardImg.setImageResource(itemList.get(position).img);
42    }
43
44    @Override
45    public int getItemCount() {
46        return itemList.size();
47    }
48
49}
```

```

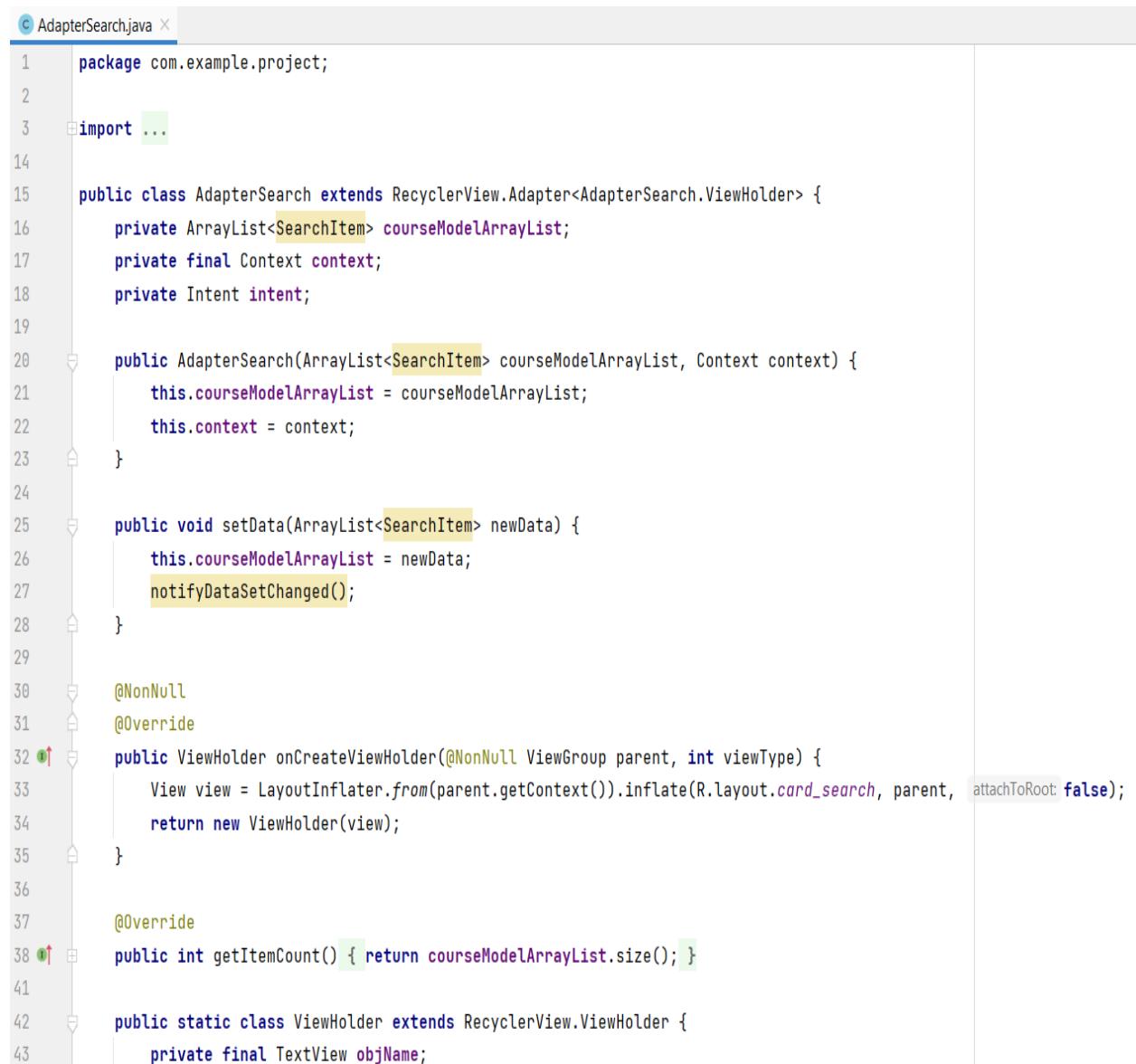
  C AdapterHome.java ×
47  @Override
48  public void onBindViewHolder(AdapterHome.MyViewHolder holder, int position) {
49      MyItem item = itemList.get(position);
50      holder.textTitle.setText(item.getTitle());
51      String image = urlImg + item.getImage();
52      Picasso.get().load(image).into(holder.cardImg);
53
54      holder.itemView.setOnClickListener(new View.OnClickListener() {
55          @Override
56          public void onClick(View view) {
57              Class<?> objectType = item.getObjectType();
58              if (objectType == location.class) {
59                  intent = new Intent(context, ExploreDetailActivity.class);
60                  location loc = (location) item.getObj();
61                  intent.putExtra( name: "object", loc);
62              }else if(objectType == ticket.class){
63                  intent = new Intent(context, TicketDetailActivity.class);
64                  ticket tix = (ticket) item.getObj();
65                  intent.putExtra( name: "object", tix);
66              }else if(objectType == hotel.class){
67                  intent = new Intent(context, HotelDetailActivity.class);
68                  hotel htl = (hotel) item.getObj();
69                  intent.putExtra( name: "object", htl);
70              }else if(objectType == tour.class){
71                  intent = new Intent(context, TourDetailActivity.class);
72                  tour trp = (tour) item.getObj();
73                  intent.putExtra( name: "object", trp);
74              }
75              context.startActivity(intent);
76          }
77      });
78  }
  C AdapterHome.java ×
80
81  @Override
82  public int getItemCount() { return itemList.size(); }
83
84

```

Kode java AdapterHome di atas merupakan implementasi dari adapter *AdapterHome* untuk *RecyclerView* dalam aplikasi Android. Adapter ini digunakan untuk menampilkan daftar item dalam tampilan kartu (*card\_home*) dengan judul dan gambar. Data item disimpan dalam *ArrayList<MyItem>* dan diinisialisasi melalui konstruktor adapter. Pada metode *onCreateViewHolder*, tampilan kartu di-inflate dan objek *MyViewHolder* dibuat untuk menampung referensi elemen UI seperti *TextView* dan *ImageView*. Pada metode *onBindViewHolder*, data dari item saat ini diatur ke dalam elemen UI dan gambar di-load menggunakan *Picasso*. Setiap item dalam daftar memiliki *OnClickListener* yang akan membuka aktivitas detail yang sesuai (*ExploreDetailActivity*, *TicketDetailActivity*, *HotelDetailActivity*, atau *TourDetailActivity*) berdasarkan tipe objek yang dikandung item tersebut. Kode ini

mengimplementasikan fitur untuk menampilkan dan mengelola daftar dinamis item dalam tampilan kartu, serta menangani navigasi ke detail item yang diklik.

### 3.2.4. AdapterSearch



```
1 package com.example.project;
2
3 import ...
4
5
6 public class AdapterSearch extends RecyclerView.Adapter<AdapterSearch.ViewHolder> {
7     private ArrayList<SearchItem> courseModelArrayList;
8     private final Context context;
9     private Intent intent;
10
11     public AdapterSearch(ArrayList<SearchItem> courseModelArrayList, Context context) {
12         this.courseModelArrayList = courseModelArrayList;
13         this.context = context;
14     }
15
16     public void setData(ArrayList<SearchItem> newData) {
17         this.courseModelArrayList = newData;
18         notifyDataSetChanged();
19     }
20
21     @NonNull
22     @Override
23     public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
24         View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.card_search, parent, attachToRoot: false);
25         return new ViewHolder(view);
26     }
27
28     @Override
29     public int getItemCount() { return courseModelArrayList.size(); }
30
31     public static class ViewHolder extends RecyclerView.ViewHolder {
32         private final TextView objName;
33
34         public ViewHolder(View itemView) {
35             super(itemView);
36             objName = itemView.findViewById(R.id.tv_obj_name);
37         }
38     }
39
40     @Override
41     public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
42         holder.objName.setText(courseModelArrayList.get(position).objName);
43     }
44 }
```

```
  AdapterSearch.java x
44      private final TextView objType;
45      public ViewHolder(@NonNull View itemView) {
46          super(itemView);
47          objName = itemView.findViewById(R.id.searchName);
48          objType = itemView.findViewById(R.id.searchType);
49      }
50  }
51
52  @Override
53  public void onBindViewHolder(@androidx.annotation.NonNull ViewHolder holder, int position) {
54      SearchItem model = courseModelArrayList.get(position);
55      holder.objName.setText(model.getTitle());
56      holder.objType.setText(model.getType());
57      holder.itemView.setOnClickListener(new View.OnClickListener() {
58          @Override
59          public void onClick(View view) {
60              Class<?> objectType = model.getObjectType();
61              if (objectType == location.class) {
62                  intent = new Intent(context, ExploreDetailActivity.class);
63                  location loc = (location) model.getObj();
64                  intent.putExtra( name: "object", loc);
65              } else if (objectType == ticket.class) {
66                  intent = new Intent(context, TicketDetailActivity.class);
67                  ticket tix = (ticket) model.getObj();
68                  intent.putExtra( name: "object", tix);
69              } else if (objectType == hotel.class) {
70                  intent = new Intent(context, HotelDetailActivity.class);
71                  hotel htl = (hotel) model.getObj();
72                  intent.putExtra( name: "object", htl);
73              } else if (objectType == tour.class) {
74                  intent = new Intent(context, TourDetailActivity.class);
```

```
  AdapterSearch.java x
75          tour trp = (tour) model.getObj();
76          intent.putExtra( name: "object", trp);
77      }
78      context.startActivity(intent);
79  });
80  }
81 }
82 }
```

Kode java AdapterSearch di atas merupakan implementasi dari adapter *AdapterSearch* untuk *RecyclerView* dalam aplikasi Android. Adapter ini digunakan untuk menampilkan hasil pencarian dalam bentuk kartu (*card\_search*) yang berisi nama dan tipe objek pencarian. Data item disimpan dalam *ArrayList<SearchItem>* dan diinisialisasi melalui konstruktor adapter. Metode *onCreateViewHolder* meng-inflate tampilan kartu dan membuat objek *ViewHolder* untuk menampung referensi elemen UI seperti *TextView*. Pada metode *onBindViewHolder*, data dari item saat ini diatur ke dalam elemen UI. Setiap item dalam daftar memiliki *OnClickListener* yang akan membuka aktivitas detail yang sesuai (*ExploreDetailActivity*, *TicketDetailActivity*, *HotelDetailActivity*, atau *TourDetailActivity*) berdasarkan tipe objek yang dikandung item tersebut. Kode ini mengimplementasikan fitur untuk menampilkan dan mengelola daftar dinamis hasil pencarian, serta menangani navigasi ke detail item yang diklik.

### 3.2.5. AdapterViewExplore

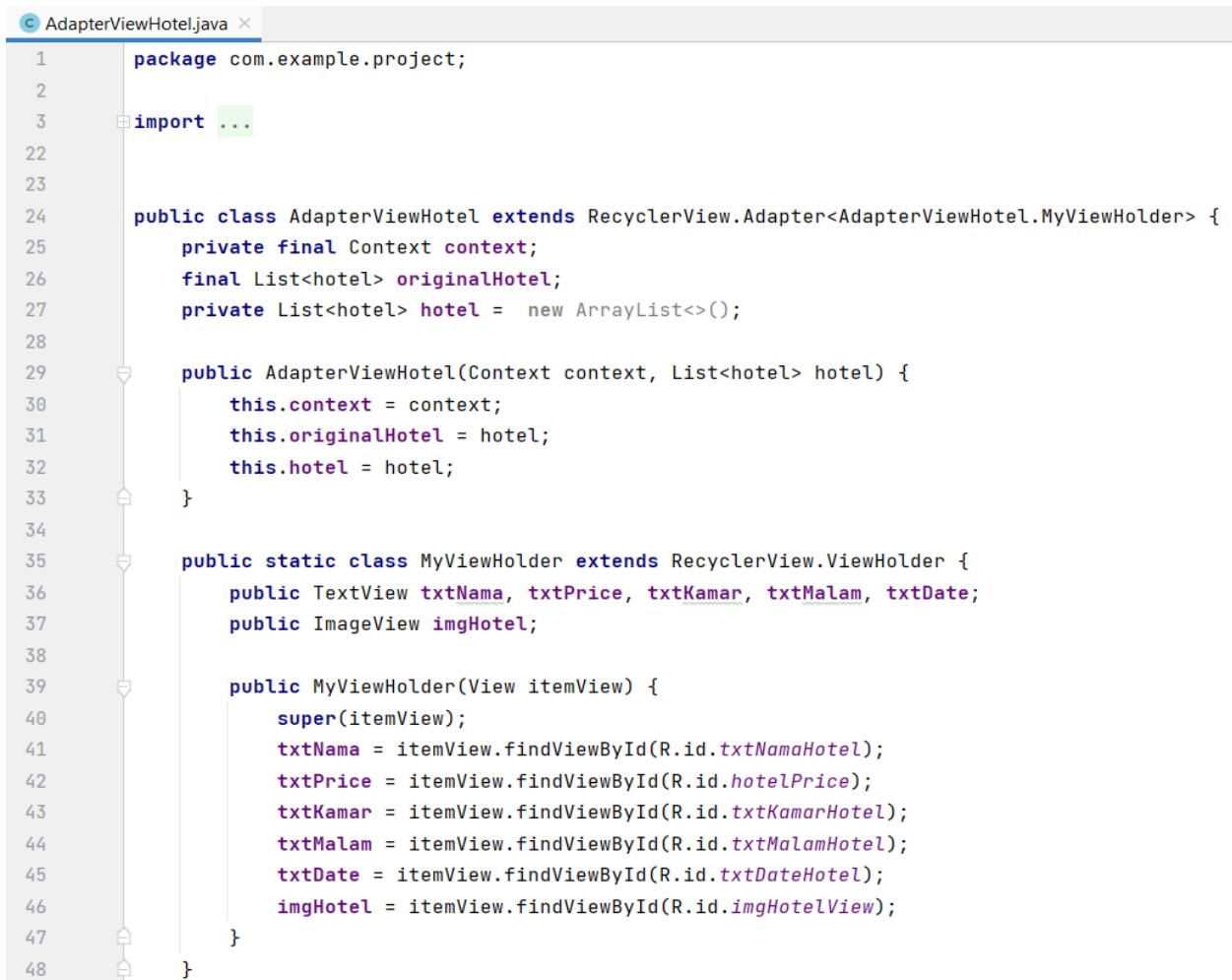
```
1 package com.example.project;
2
3 import ...
4
5
6 public class AdapterViewExplore extends RecyclerView.Adapter<AdapterViewExplore.ViewHolder> implements Filterable {
7     private final Context context;
8     private ArrayList<ExploreData> exploreList;
9     private ArrayList<ExploreData> exploreListFull;
10    private String urlImg = "http://192.168.100.4/ezgo/images/";
11
12    public AdapterViewExplore(Context context, ArrayList<ExploreData> exploreList) {
13        this.context = context;
14        this.exploreList = exploreList;
15        this.exploreListFull = new ArrayList<>(exploreList); // copy data asli
16    }
17
18    public void setExploreDataList(ArrayList<ExploreData> exploreList) {
19        this.exploreList = exploreList;
20        this.exploreListFull = new ArrayList<>(exploreList); // copy data asli
21        notifyDataSetChanged();
22    }
23
24    public static class ViewHolder extends RecyclerView.ViewHolder {
25        public TextView txtPlace, txtRating;
26        public ImageView imgDis;
27
28        public ViewHolder(View itemView) {
29            super(itemView);
30            txtPlace = itemView.findViewById(R.id.txtPlace);
31            imgDis = itemView.findViewById(R.id.imgDis);
32            txtRating = itemView.findViewById(R.id.txtDesc);
33        }
34    }
35
36    @Override
37    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
38        View itemView = LayoutInflater.from(context).inflate(R.layout.item_explore, parent, false);
39        return new ViewHolder(itemView);
40    }
41
42    @Override
43    public void onBindViewHolder(ViewHolder holder, int position) {
44        ExploreData exploreData = exploreListFull.get(position);
45        holder.txtPlace.setText(exploreData.getPlace());
46        holder.txtRating.setText(exploreData.getRating());
47        holder.imgDis.setImageResource(exploreData.getImgDis());
48    }
49
50    @Override
51    public int getItemCount() {
52        return exploreListFull.size();
53    }
54
55    @Override
56    public void onBindViewHolder(ViewHolder holder, int position, List payloads) {
57        super.onBindViewHolder(holder, position, payloads);
58    }
59
60    @Override
61    public void notifyDataSetChanged() {
62        super.notifyDataSetChanged();
63    }
64
65    @Override
66    public Filter getFilter() {
67        return null;
68    }
69
70    @Override
71    public void onAttachedToRecyclerView(RecyclerView recyclerView) {
72        super.onAttachedToRecyclerView(recyclerView);
73    }
74
75    @Override
76    public void onDetachedFromRecyclerView(RecyclerView recyclerView) {
77        super.onDetachedFromRecyclerView(recyclerView);
78    }
79
80    @Override
81    public void onScrolled(RecyclerView recyclerView, int dx, int dy) {
82        super.onScrolled(recyclerView, dx, dy);
83    }
84
85    @Override
86    public void onItemRangeChanged(int positionStart, int itemCount) {
87        super.onItemRangeChanged(positionStart, itemCount);
88    }
89
90    @Override
91    public void onItemRangeChanged(int positionStart, int itemCount, Object payload) {
92        super.onItemRangeChanged(positionStart, itemCount, payload);
93    }
94
95    @Override
96    public void onItemRangeInserted(int positionStart, int itemCount) {
97        super.onItemRangeInserted(positionStart, itemCount);
98    }
99
100   @Override
101  public void onItemRangeRemoved(int positionStart, int itemCount) {
102      super.onItemRangeRemoved(positionStart, itemCount);
103  }
104
105  @Override
106  public void onItemRangeChanged(int positionStart, int itemCount, Object payload, int payloadType) {
107      super.onItemRangeChanged(positionStart, itemCount, payload, payloadType);
108  }
109
110 }
```

```
AdapterViewExplore.java ×
49     }
50 }
51
52     @NonNull
53     @Override
54     public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
55         View view = LayoutInflater.from(context).inflate(R.layout.card_explore, parent, attachToRoot: false);
56         return new ViewHolder(view);
57     }
58
59     @Override
60     public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
61         ExploreData exploreData = exploreList.get(position);
62         holder.txtPlace.setText(exploreData.getTitle());
63         holder.txtRating.setText(exploreData.getRating() + " Likes");
64         String img = urlImg + exploreData.getImage();
65         Picasso.get().load(img).into(holder.imgDis);
66         holder.itemView.setOnClickListener(view -> {
67             Intent i = new Intent(context, ExploreDetailActivity.class);
68             location loc = exploreData.getObj();
69             i.putExtra( name: "object", loc);
70             context.startActivity(i);
71         });
72     }
73
74     @Override
75     public int getItemCount() { return exploreList.size(); }
76
77     @Override
78     public Filter getFilter() { return exploreFilter; }
79
80 }
```

```
AdapterViewExplore.java ×
84     private Filter exploreFilter = new Filter() {
85         @Override
86         protected FilterResults performFiltering(CharSequence constraint) {
87             List<ExploreData> filteredList = new ArrayList<>();
88
89             if (constraint == null || constraint.length() == 0) {
90                 filteredList.addAll(exploreListFull);
91             } else {
92                 String filterPattern = constraint.toString().toLowerCase().trim();
93
94                 for (ExploreData item : exploreListFull) {
95                     if (item.getTitle().toLowerCase().contains(filterPattern)) {
96                         filteredList.add(item);
97                     }
98                 }
99             }
100
101             FilterResults results = new FilterResults();
102             results.values = filteredList;
103             return results;
104         }
105
106         @Override
107         protected void publishResults(CharSequence constraint, FilterResults results) {
108             exploreList.clear();
109             exploreList.addAll((List) results.values);
110             notifyDataSetChanged();
111         }
112     };
113 }
```

Kode java AdapterViewExplore di atas merupakan implementasi dari adapter *AdapterViewExplore* untuk *RecyclerView* dalam aplikasi Android yang menampilkan daftar tempat eksplorasi. Adapter ini mengelola daftar data *ExploreData*, menampilkan setiap item dalam tampilan kartu (*card\_explore*) yang berisi judul tempat, rating (jumlah likes), dan gambar. Gambar di-load menggunakan *Picasso* dari URL yang ditentukan. Adapter ini juga mengimplementasikan fitur filter untuk memungkinkan pengguna mencari tempat berdasarkan judul. Data yang difilter ditampilkan dengan memodifikasi daftar data asli (*exploreListFull*). Ketika item diklik, intent akan membuka *ExploreDetailActivity* dan mengirimkan objek lokasi yang dipilih. Kode ini mengimplementasikan fitur untuk menampilkan, memfilter, dan mengelola daftar tempat eksplorasi, serta menangani navigasi ke detail tempat yang dipilih.

### 3.2.6. AdapterViewHotel



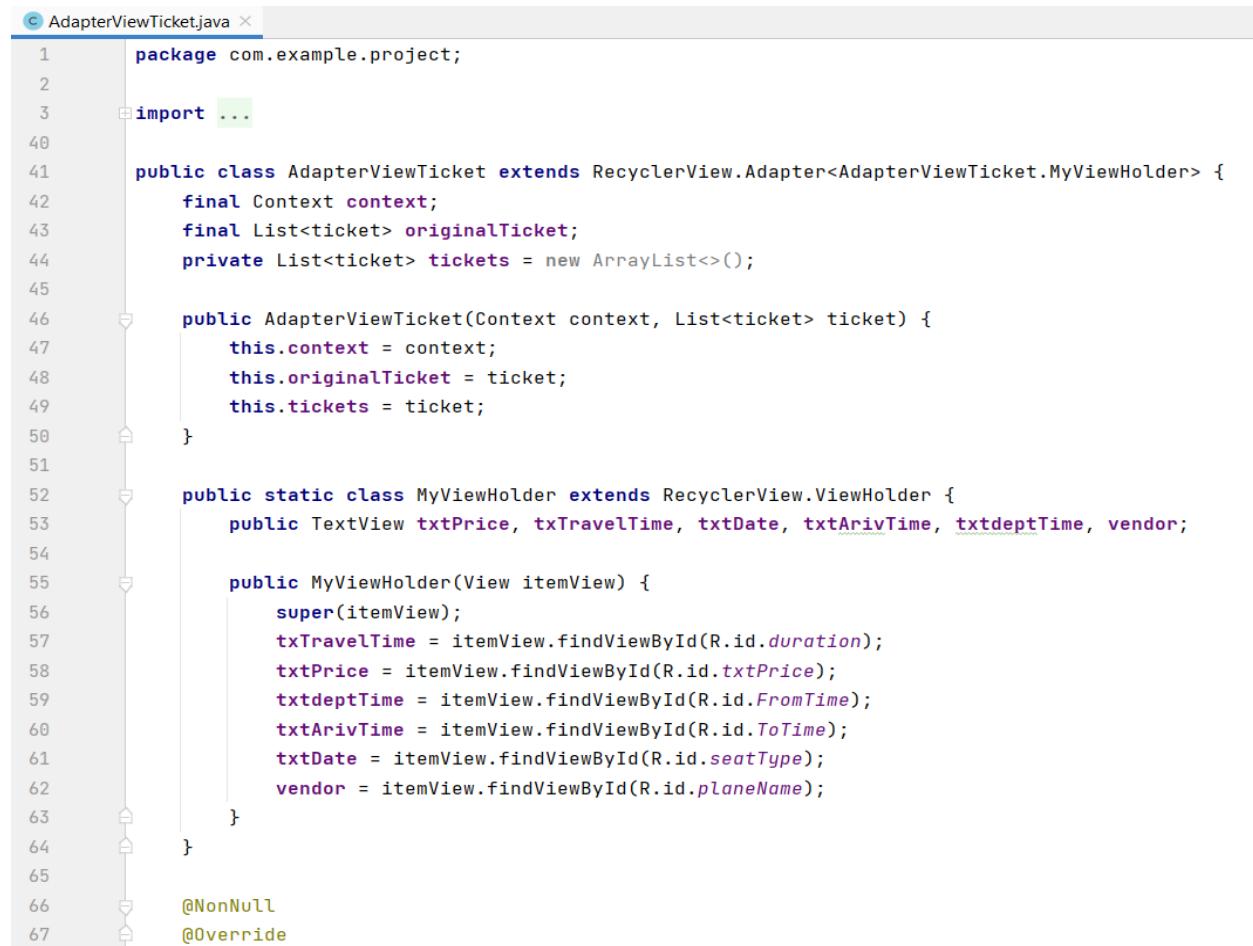
```
c AdapterViewHotel.java x
1 package com.example.project;
2
3 import ...
4
5
6 public class AdapterViewHotel extends RecyclerView.Adapter<AdapterViewHotel.MyViewHolder> {
7     private final Context context;
8     final List<hotel> originalHotel;
9     private List<hotel> hotel = new ArrayList<>();
10
11     public AdapterViewHotel(Context context, List<hotel> hotel) {
12         this.context = context;
13         this.originalHotel = hotel;
14         this.hotel = hotel;
15     }
16
17     public static class MyViewHolder extends RecyclerView.ViewHolder {
18         public TextView txtNama, txtPrice, txtKamar, txtMalam, txtDate;
19         public ImageView imgHotel;
20
21         public MyViewHolder(View itemView) {
22             super(itemView);
23             txtNama = itemView.findViewById(R.id.txtNamaHotel);
24             txtPrice = itemView.findViewById(R.id.hotelPrice);
25             txtKamar = itemView.findViewById(R.id.txtKamarHotel);
26             txtMalam = itemView.findViewById(R.id.txtMalamHotel);
27             txtDate = itemView.findViewById(R.id.txtDateHotel);
28             imgHotel = itemView.findViewById(R.id.imgHotelView);
29         }
30     }
31 }
```

```
c AdapterViewHotel.java ×
50     @NotNull
51
52     @Override
53     public AdapterViewHotel.MyViewHolder onCreateViewHolder(@NotNull ViewGroup parent, int viewType) {
54         View view = LayoutInflater.from(context).inflate(R.layout.card_hotel, parent, attachToRoot: false);
55         return new MyViewHolder(view);
56     }
57
58     @Override
59     public void onBindViewHolder(AdapterViewHotel.MyViewHolder holder, int position) {
60         hotel hotelData = hotel.get(position);
61
62         DecimalFormat decimalFormat = new DecimalFormat( pattern: "#,###");
63         String price = decimalFormat.format(hotelData.hPrice) + "/Pax";
64
65         String nights = hotelData.hNights + " Nights Available";
66
67         holder.txtNama.setText(hotelData.hName);
68         holder.txtPrice.setText(price);
69         holder.txtKamar.setText(hotelData.hRoomType);
70         holder.txtMalam.setText(nights);
71
72         SimpleDateFormat inputFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
73
74         Date dateForm = null;
75         try {
76             dateForm = inputFormat.parse(hotelData.hDate);
77         } catch (ParseException e) {
78             throw new RuntimeException(e);
79         }
80     }
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100 }
```

```
c AdapterViewHotel.java ×
80         SimpleDateFormat outputFormat = new SimpleDateFormat( pattern: "dd MMMM yyyy");
81         String outputDateString = outputFormat.format(dateForm);
82
83         holder.txtDate.setText(outputDateString);
84
85         String urlImg = "http://192.168.100.4/ezgo/images/";
86         String image = urlImg + hotelData.hImage;
87
88         Picasso.get().load(image).into(holder.imgHotel);
89
90         holder.itemView.setOnClickListener(view -> {
91             Intent i = new Intent(context, HotelDetailActivity.class);
92             i.putExtra( name: "object", hotelData);
93             context.startActivity(i);
94         });
95     }
96
97     @Override
98     public int getItemCount() { return hotel.size(); }
99
100 }
```

Kode java AdapterViewHotel di atas merupakan implementasi dari adapter *AdapterViewHotel* untuk *RecyclerView* dalam aplikasi Android yang digunakan untuk menampilkan daftar hotel. Adapter ini menerima konteks dan daftar objek *hotel*, menampilkan setiap item dalam tampilan kartu (*card\_hotel*) yang berisi informasi nama hotel, harga, jenis kamar, jumlah malam tersedia, dan tanggal. Gambar hotel di-load menggunakan *Picasso* dari URL yang ditentukan. Adapter ini mengatur format tampilan harga dan tanggal, serta mengubah format tanggal dari "yyyy-MM-dd" ke "dd MMMM yyyy". Saat item hotel diklik, intent akan membuka *HotelDetailActivity* dan mengirimkan objek hotel yang dipilih. Kode ini mengimplementasikan fitur untuk menampilkan dan mengelola daftar hotel, serta menangani navigasi ke detail hotel yang dipilih, meningkatkan pengalaman pengguna dalam menjelajahi informasi hotel.

### 3.2.7. AdapterViewTicket



```
1 package com.example.project;
2
3 import ...
4
5
6 public class AdapterViewTicket extends RecyclerView.Adapter<AdapterViewTicket.MyViewHolder> {
7     final Context context;
8     final List<ticket> originalTicket;
9     private List<ticket> tickets = new ArrayList<>();
10
11     public AdapterViewTicket(Context context, List<ticket> ticket) {
12         this.context = context;
13         this.originalTicket = ticket;
14         this.tickets = ticket;
15     }
16
17     public static class MyViewHolder extends RecyclerView.ViewHolder {
18         public TextView txtPrice, txTravelTime, txtDate, txtArrivTime, txtdeptTime, vendor;
19
20         public MyViewHolder(View itemView) {
21             super(itemView);
22             txTravelTime = itemView.findViewById(R.id.duration);
23             txtPrice = itemView.findViewById(R.id.txtPrice);
24             txtdeptTime = itemView.findViewById(R.id.FromTime);
25             txtArrivTime = itemView.findViewById(R.id.ToTime);
26             txtDate = itemView.findViewById(R.id.seatType);
27             vendor = itemView.findViewById(R.id.planeName);
28         }
29     }
30
31     @NonNull
32     @Override
```

```
© AdapterViewTicket.java ×
67     @Override
68     public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
69         View view = LayoutInflater.from(context).inflate(R.layout.card_ticket, parent, attachToRoot: false);
70         return new MyViewHolder(view);
71     }
72
73     @Override
74     public void onBindViewHolder(MyViewHolder holder, int position) {
75         ticket tix = tickets.get(position);
76
77         DecimalFormat decimalFormat = new DecimalFormat( pattern: "#,###");
78         String price = "Rp "+decimalFormat.format(tix.tcPrice)+"/Pax";
79
80         holder.txTravelTime.setText(tix.tcTravelTime);
81         holder.txtDeptTime.setText(tix.tcDepartureTime);
82
83         DateTimeFormatter formatter = null;
84         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
85             formatter = DateTimeFormatter.ofPattern("HH:mm:ss");
86         }
87
88         LocalTime sum = null;
89         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
90             LocalTime time1 = LocalTime.parse(tix.tcDepartureTime, formatter);
91             LocalTime time2 = LocalTime.parse(tix.tcTravelTime, formatter);
92
93             sum = time1.plusHours(time2.getHour())
94                 .plusMinutes(time2.getMinute())
95                 .plusSeconds(time2.getSecond());
96         }
97     }
98 }
```

```
© AdapterViewTicket.java ×
98     SimpleDateFormat inputFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
99
100    Date dateForm = null;
101    try {
102        dateForm = inputFormat.parse(tix.tcDate);
103    } catch (ParseException e) {
104        throw new RuntimeException(e);
105    }
106
107    SimpleDateFormat outputFormat = new SimpleDateFormat( pattern: "dd MMMM yyyy");
108    String outputDateString = outputFormat.format(dateForm);
109
110    holder.txtDate.setText(outputDateString);
111    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
112        holder.txtArrivTime.setText(sum.format(formatter));
113    }
114    holder.txtPrice.setText(price);
115    holder.itemView.setOnClickListener(view -> {
116        Intent i = new Intent(context, TicketDetailActivity.class);
117        i.putExtra( name: "object", tix);
118        context.startActivity(i);
119    });
120 }
121
122     @Override
123     public int getItemCount() { return originalTicket.size(); }
124 }
```

Kode java AdapterViewTicket di atas merupakan implementasi dari adapter *AdapterViewTicket* untuk *RecyclerView* dalam aplikasi Android yang digunakan untuk menampilkan daftar tiket. Adapter ini menerima konteks dan daftar objek *ticket*, menampilkan setiap item dalam tampilan kartu (*card\_ticket*) yang berisi informasi harga, waktu perjalanan, waktu keberangkatan, waktu tiba, tanggal, dan nama vendor. Kode ini menggunakan *DecimalFormat* untuk format harga dan *SimpleDateFormat* untuk mengubah format tanggal dari "yyyy-MM-dd" ke "dd MMMM yyyy". Untuk menghitung waktu tiba, kode ini menggunakan *DateTimeFormatter* dan *LocalTime* (jika SDK versi O atau lebih tinggi). Ketika item tiket diklik, intent akan membuka *TicketDetailActivity* dan mengirimkan objek tiket yang dipilih. Kode ini mengimplementasikan fitur untuk menampilkan dan mengelola daftar tiket, serta menangani navigasi ke detail tiket yang dipilih, sehingga pengguna dapat melihat informasi lengkap tentang tiket yang tersedia.

### 3.2.8. AdapterViewTour

```

  AdapterViewTour.java ×
1  package com.example.project;
2
3  import ...
18
19  public class AdapterViewTour extends RecyclerView.Adapter<AdapterViewTour.MyViewHolder> {
20      private final Context context;
21      final List<tour> originalTour;
22      private List<tour> tour = new ArrayList<>();
23
24      public AdapterViewTour(Context context, List<tour> tour) {
25          this.context = context;
26          this.originalTour = tour;
27          this.tour = tour;
28      }
29
30      public static class MyViewHolder extends RecyclerView.ViewHolder {
31          public TextView textTitle;
32          public ImageView imgTour;
33
34          public MyViewHolder(View itemView) {
35              super(itemView);
36              textTitle = itemView.findViewById(R.id.tourNameView);
37              imgTour = itemView.findViewById(R.id.imgTourView);
38          }
39      }
40
41      @NonNull
42      @Override
43      public AdapterViewTour.MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
44          View view = LayoutInflater.from(context).inflate(R.layout.card_tour, parent, attachToRoot: false);
45          return new MyViewHolder(view);

```

```
  C AdapterViewTour.java x
46
47
48
49  @Override
50  public void onBindViewHolder(AdapterViewTour.MyViewHolder holder, int position) {
51
52      tour trp = tour.get(position);
53
54      holder.textTitle.setText(trp.tpName);
55
56      String urlImg = "http://192.168.100.4/ezgo/images/";
57      String image = urlImg + trp.tpImage;
58
59      Picasso.get().load(image).into(holder.imgTour);
60
61      holder.itemView.setOnClickListener(view -> {
62          Intent i = new Intent(context, TourDetailActivity.class);
63          i.putExtra("name: "object", trp);
64          context.startActivity(i);
65      });
66
67  @Override
68  public int getItemCount() { return tour.size(); }
69 }
```

Kode java AdapterViewTour di atas merupakan implementasi dari adapter *AdapterViewTour* untuk *RecyclerView* dalam aplikasi Android yang digunakan untuk menampilkan daftar paket tour. Adapter ini menerima konteks dan daftar objek *tour*, menampilkan setiap item dalam tampilan kartu (*card\_tour*) yang berisi judul tour dan gambar. Gambar di-load menggunakan *Picasso* dari URL yang ditentukan. Ketika item tour diklik, intent akan membuka *TourDetailActivity* dan mengirimkan objek tour yang dipilih. Adapter ini mengelola daftar data tour asli dan menampilkan informasi masing-masing tour dalam tampilan kartu, serta menyediakan fitur navigasi ke detail tour yang dipilih, sehingga pengguna dapat melihat informasi lengkap tentang tour yang tersedia. Kode ini mengimplementasikan fitur untuk menampilkan dan mengelola daftar paket tour, meningkatkan pengalaman pengguna dalam menjelajahi informasi tour.

### 3.2.9. ChangePassActivity

```
ChangePassActivity.java
1 package com.example.project;
2
3 import ...
25
26 public class ChangePassActivity extends AppCompatActivity {
27     TextInputEditText inputUsername, inputNewPass;
28     SharedPreferences preferences;
29     User user;
30
31     @Override
32     protected void onCreate(Bundle savedInstanceState) {
33         super.onCreate(savedInstanceState);
34         setContentView(R.layout.activity_change_pass);
35
36         inputUsername = findViewById(R.id.inputUsername);
37         inputNewPass = findViewById(R.id.inputNewPass);
38
39         preferences = getSharedPreferences("ezgo", MODE_PRIVATE);
40         String userJson = preferences.getString("user", null);
41         user = new Gson().fromJson(userJson, User.class);
42
43         findViewById(R.id.btnSwave).setOnClickListener(new View.OnClickListener() {
44             @Override
45             public void onClick(View view) {
46                 String username = inputUsername.getText().toString().trim();
47                 String newPass = inputNewPass.getText().toString().trim();
48
49                 if (username.isEmpty() || newPass.isEmpty()) {
50                     Toast.makeText(context: ChangePassActivity.this, text: "Please fill in all fields", Toast.LENGTH_SHORT).show();
51                     return;
52                 }
53             }
54
55         });
56     }
57 }
58
59 private void updatePassword(String username, String newPassword) {
60     RequestQueue queue = Volley.newRequestQueue(context: this);
61     Gson gson = new Gson();
62
63     Map<String, Object> params = new HashMap<>();
64     params.put("controller", "account");
65     params.put("method", "changePassword");
66     params.put("userID", user.userID);
67     params.put("username", username);
68     params.put("newPassword", newPassword);
69
70     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, url: "http://192.168.100.4/ezgo/router.php",
71         new JSONObject(params),
72         new Response.Listener<JSONObject>() {
73             @Override
74             public void onResponse(JSONObject response) {
75                 Log.d(tag: "ChangePassActivity", msg: "Response: " + response);
76
77                 // Handle response
78                 boolean success = true;
79                 if (success) {
80                     Toast.makeText(context: ChangePassActivity.this, text: "Password updated successfully", Toast.LENGTH_SHORT).show();
81                     finish(); // Close activity after successful update
82                 } else {
83                     Toast.makeText(context: ChangePassActivity.this, text: "Failed to update password", Toast.LENGTH_SHORT).show();
84                 }
85             }
86         }
87     );
88     queue.add(jsonObjectRequest);
89 }
```

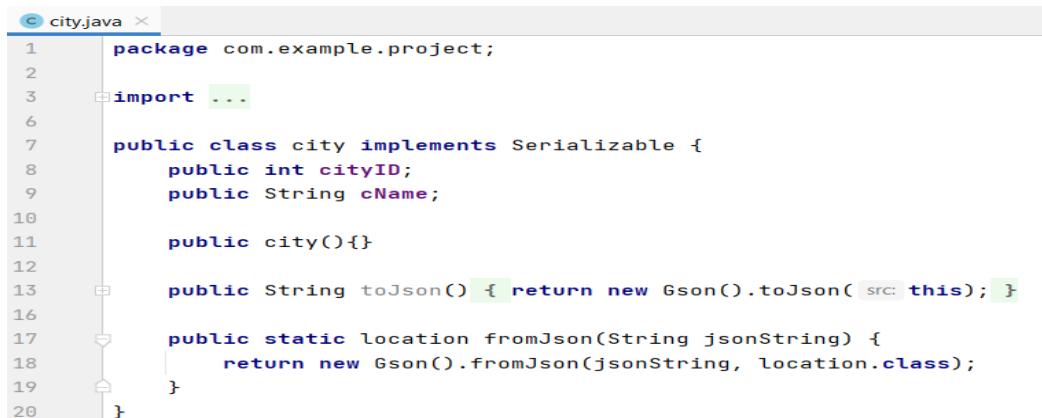
```
ChangePassActivity.java
53     updatePassword(username, newPassword);
54
55 }
56 }
57 }
58
59 private void updatePassword(String username, String newPassword) {
60     RequestQueue queue = Volley.newRequestQueue(context: this);
61     Gson gson = new Gson();
62
63     Map<String, Object> params = new HashMap<>();
64     params.put("controller", "account");
65     params.put("method", "changePassword");
66     params.put("userID", user.userID);
67     params.put("username", username);
68     params.put("newPassword", newPassword);
69
70     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, url: "http://192.168.100.4/ezgo/router.php",
71         new JSONObject(params),
72         new Response.Listener<JSONObject>() {
73             @Override
74             public void onResponse(JSONObject response) {
75                 Log.d(tag: "ChangePassActivity", msg: "Response: " + response);
76
77                 // Handle response
78                 boolean success = true;
79                 if (success) {
80                     Toast.makeText(context: ChangePassActivity.this, text: "Password updated successfully", Toast.LENGTH_SHORT).show();
81                     finish(); // Close activity after successful update
82                 } else {
83                     Toast.makeText(context: ChangePassActivity.this, text: "Failed to update password", Toast.LENGTH_SHORT).show();
84                 }
85             }
86         }
87     );
88     queue.add(jsonObjectRequest);
89 }
```



```
84     }
85     }
86     }
87     new Response.ErrorListener() {
88         @Override
89         public void onErrorResponse(VolleyError error) {
90             Log.e("ChangePassActivity", "Error: " + error.toString());
91             if (error.networkResponse != null) {
92                 String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
93                 Log.e("ChangePassActivity", "Response: " + responseBody);
94             }
95             Toast.makeText(context, "Failed to update password", Toast.LENGTH_SHORT).show();
96         }
97     });
98     queue.add(jsonObjectRequest);
99 }
100 }
```

Kode java *ChangePassActivity* di atas merupakan implementasi dari aktivitas *ChangePassActivity* dalam aplikasi Android yang digunakan untuk mengubah kata sandi pengguna. Saat aktivitas ini dibuat, elemen UI seperti *TextInputEditText* untuk username dan kata sandi baru diinisialisasi, serta *SharedPreferences* untuk membaca data pengguna yang tersimpan. Ketika tombol simpan (*btnSvave*) diklik, metode *updatePassword* dipanggil untuk mengirim permintaan HTTP POST menggunakan *Volley* ke server dengan URL *router.php*. Data yang dikirimkan mencakup *controller*, *method*, *userID*, *username*, dan *newPassword*. Jika permintaan berhasil, pesan konfirmasi ditampilkan dan aktivitas ditutup; jika gagal, pesan kesalahan ditampilkan. Kode ini mengimplementasikan fitur untuk memungkinkan pengguna mengubah kata sandi mereka dengan mengirimkan data yang diperbarui ke server dan menampilkan hasilnya melalui antarmuka pengguna.

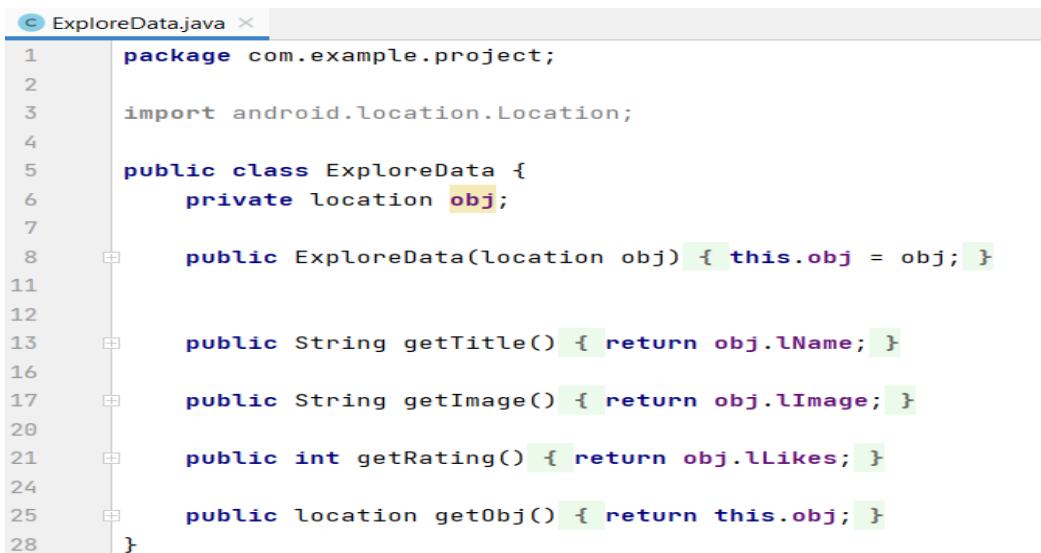
### 3.2.10. city



```
1 package com.example.project;
2
3 import ...
4
5
6 public class city implements Serializable {
7     public int cityID;
8     public String cName;
9
10    public city(){}
11
12    public String toJson() { return new Gson().toJson(this); }
13
14    public static location fromJson(String jsonString) {
15        return new Gson().fromJson(jsonString, location.class);
16    }
17
18
19 }
20 }
```

Kode java city di atas merupakan implementasi dari kelas *city* dalam aplikasi Android yang digunakan untuk merepresentasikan objek kota. Kelas ini mengimplementasikan antarmuka *Serializable*, sehingga objek *city* dapat dengan mudah diserialisasi dan dideserialisasi. Kelas ini memiliki dua atribut: *cityID* (ID kota) dan *cName* (nama kota). Konstruktur default (*city()*) disediakan, serta dua metode: *toJson()* untuk mengonversi objek *city* menjadi string JSON menggunakan Gson, dan *fromJson(String jsonString)* untuk membuat objek *location* dari string JSON. Metode *fromJson* seharusnya mengembalikan objek *city* bukan *location*. Kode ini mengimplementasikan fitur serialisasi dan deserialisasi JSON yang memungkinkan pertukaran data objek *city* dalam format JSON, yang sangat berguna untuk komunikasi data antar bagian aplikasi atau antara aplikasi dan server.

### 3.2.11. ExploreData



```
1 package com.example.project;
2
3 import android.location.Location;
4
5 public class ExploreData {
6     private location obj;
7
8     public ExploreData(location obj) { this.obj = obj; }
11
12
13     public String getTitle() { return obj.lName; }
16
17     public String getImage() { return obj.lImage; }
20
21     public int getRating() { return obj.lLikes; }
24
25     public location getObj() { return this.obj; }
28 }
```

Kode java ExploreData di atas merupakan implementasi dari kelas *ExploreData* dalam aplikasi Android yang digunakan untuk merepresentasikan data eksplorasi. Kelas ini memiliki atribut *obj* yang merupakan objek dari kelas *location*. Konstruktur kelas ini menerima objek *location* dan menginisialisasi atribut *obj* dengannya. Kelas ini menyediakan metode *getTitle()*, *getImage()*, dan *getRating()* untuk mengembalikan nama, gambar, dan jumlah likes dari lokasi, serta metode *getObj()* untuk mengembalikan objek *location* itu sendiri. Kode ini mengimplementasikan fitur encapsulasi data lokasi yang memungkinkan penyimpanan dan pengambilan informasi lokasi dengan mudah dalam konteks aplikasi, seperti menampilkan daftar lokasi yang dapat dieksplorasi oleh pengguna.

### 3.2.12. ExploreDetailActivity

```
ExploreDetailActivity.java
1 package com.example.project;
2
3 import ...
4
5 public class ExploreDetailActivity extends AppCompatActivity {
6     private ImageView locImg;
7     private TextView locName, locCity, locDesc, locLike;
8     private ToggleButton btnLike;
9     private MaterialButton btnWatch;
10    private int likes;
11    private LatLng position;
12
13    @Override
14    protected void onCreate(Bundle savedInstanceState) {
15        super.onCreate(savedInstanceState);
16        setContentView(R.layout.activity_explore_detail);
17        Intent intent = getIntent();
18        location loc = (location) intent.getSerializableExtra("object");
19
20        String urlReq = "http://192.168.100.4/ezgo/router.php";
21        String urlImg = "http://192.168.100.4/ezgo/images/";
22
23        FrameLayout btnBack = findViewById(R.id.backDetailExplore);
24        locImg = findViewById(R.id.imgExploreDetail);
25        btnLike = findViewById(R.id.btnLike);
26        locName = findViewById(R.id.titleExplore);
27        locCity = findViewById(R.id.address);
28        locDesc = findViewById(R.id.descExplore);
29        locLike = findViewById(R.id.locLike);
30        btnWatch = findViewById(R.id.btnWatch);
31        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.theMap);
32    }
33}
```

```
ExploreDetailActivity.java
70     RequestQueue queue = Volley.newRequestQueue(context: this);
71     Gson gson = new Gson();
72
73     Map<String, Object> params = new HashMap<>();
74     params.put("controller", "city");
75     params.put("method", "getCity");
76     params.put("cityID", loc.cityID);
77
78     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
79         new JSONObject(params),
80         response -> {
81             Log.d("Ezgo", "ResponseHome: " + response);
82             ResponseOneObject<String> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObject<String>>() {
83                 .getType());
84             boolean success = resp.isSuccess();
85             String cName = resp.getData();
86
87             if (success) {
88                 try {
89                     locName.setText(loc.lName);
90                     locCity.setText(cName);
91                     locDesc.setText(loc.lDesc);
92                     likes = loc.llLikes;
93                     locLike.setText(String.valueOf(loc.llLikes));
94                     String image = urlImg + loc.lImage;
95                     Picasso.get().load(image).into(locImg);
96                 } catch (Exception e) {
97                     Log.e("Ezgo", "Error: " + e);
98                 }
99             }
100        },
101    );
102}
```

```
ExploreDetailActivity.java
101     error -> {
102         Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
103         String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
104         Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
105     });
106     queue.add(jsonObjectRequest);
107
108     try {
109         mapFragment.getMapAsync(googleMap -> {
110             position = new LatLng(loc.llLat, loc.llLong);
111             googleMap.addMarker(new MarkerOptions().position(position).title("My Location Now")).showInfoWindow();
112             googleMap.animateCamera(CameraUpdateFactory.newLatLng(position));
113             googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(position, zoom: 16));
114         });
115     } catch (Exception e) {
116         Toast.makeText(getApplicationContext(), e.toString(), Toast.LENGTH_LONG).show();
117     }
118
119     btnBack.setOnClickListener(view -> onBackPressed());
120     btnLike.setOnCheckedChangeListener((compoundButton, b) -> {
121         if (b) {
122             likes++;
123         } else {
124             likes--;
125         }
126
127         RequestQueue queue1 = Volley.newRequestQueue( context: ExploreDetailActivity.this);
128         Gson gson1 = new Gson();
129
130         Map<String, Object> params1 = new HashMap<>();
131         params1.put( k: "controller", v: "location");

```

```
ExploreDetailActivity.java
132         params1.put( k: "method", v: "like");
133         params1.put( k: "locID", loc.locID);
134         params1.put( k: "llikes", likes);
135
136         JsonObjectRequest jsonObjectRequest1 = new JsonObjectRequest(Request.Method.POST, urlReq,
137             new JSONObject(params1),
138             response -> {
139                 Log.d( tag: "Ezgo", msg: "ResponseHome: " + response);
140                 ResponseNoObject resp = gson1.fromJson(response.toString(), new TypeToken<ResponseNoObject>() {
141                     .getType());
142                 boolean success = resp.isSuccess();
143
144                 if (success) {
145                     try {
146                         locLike.setText(String.valueOf(likes));
147                         Log.d( tag: "Ezgo", msg: "Likes: " + likes);
148                     } catch (Exception e) {
149                         Log.e( tag: "Ezgo", msg: "Error: " + e);
150                     }
151                 }
152             },
153             error -> {
154                 Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
155                 String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
156                 Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
157             });
158             queue1.add(jsonObjectRequest1);
159         });
160         btnWatch.setOnClickListener(view -> {
161             Intent intent1 = new Intent( packageContext: ExploreDetailActivity.this, WebViewActivity.class);
162             intent1.putExtra( name: "link", loc.llLink);

```

```

163     }  

164     }  

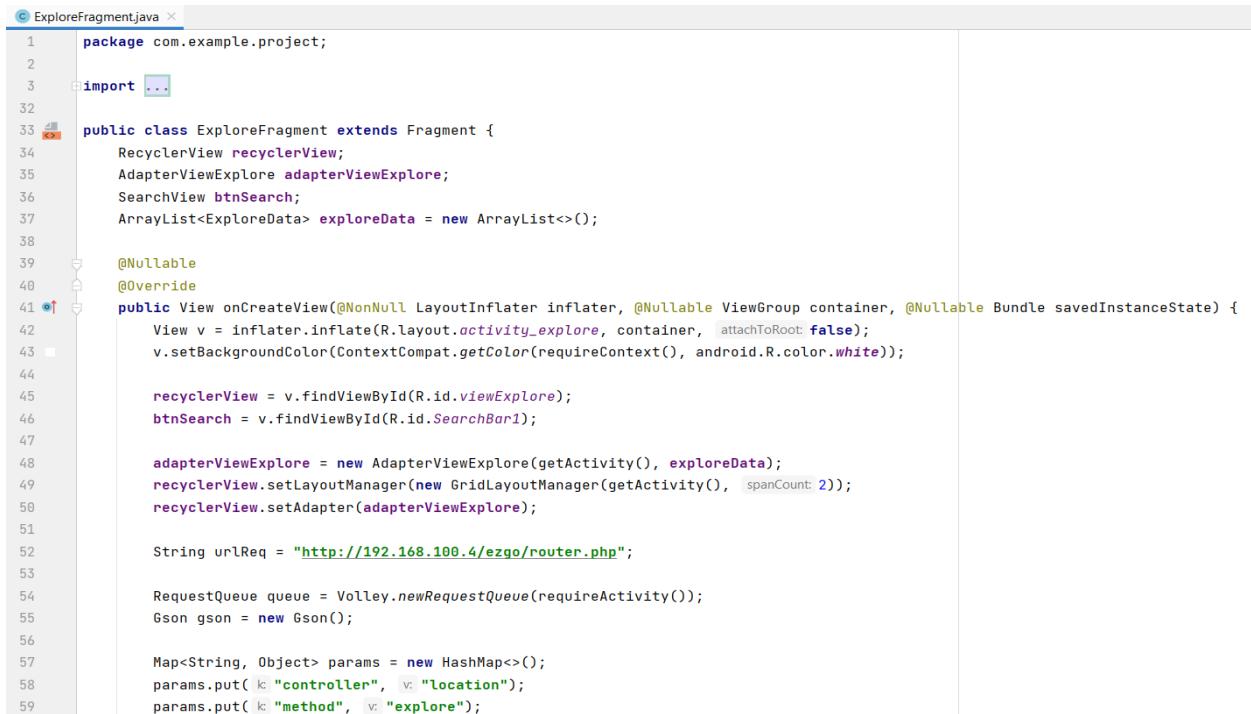
165     }  

166 }
    } ;
    startActivity(intent1);
}

```

Kode java ExploreDetailActivity di atas merupakan implementasi dari *ExploreDetailActivity* dalam aplikasi Android yang digunakan untuk menampilkan detail suatu lokasi eksplorasi. Saat aktivitas dibuat, elemen UI seperti *ImageView*, *TextView*, *ToggleButton*, dan *MaterialButton* diinisialisasi. Aktivitas ini menerima objek *location* dari intent dan menampilkan detailnya seperti nama, kota, deskripsi, dan gambar menggunakan *Picasso*. Informasi kota diambil dari server menggunakan *Volley* dan ditampilkan. Lokasi juga ditampilkan di peta menggunakan *Google Maps*. Tombol suka (*btnLike*) memungkinkan pengguna untuk menambah atau mengurangi jumlah suka, yang kemudian diperbarui di server. Tombol tonton (*btnWatch*) membuka *WebViewActivity* untuk melihat link terkait lokasi. Kode ini mengimplementasikan fitur untuk menampilkan detail lokasi eksplorasi dengan peta interaktif, serta memungkinkan pengguna untuk menyukai lokasi dan melihat informasi tambahan melalui link.

### ***3.2.13. ExploreFragmant***



```

ExploreFragment.java x
1 package com.example.project;
2
3 import ...
4
5
6 public class ExploreFragment extends Fragment {
7     RecyclerView recyclerView;
8     AdapterViewExplore adapterViewExplore;
9     SearchView btnSearch;
10    ArrayList<ExploreData> exploreData = new ArrayList<>();
11
12    @Nullable
13    @Override
14    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
15        View v = inflater.inflate(R.layout.activity_explore, container, false);
16        v.setBackgroundColor(ContextCompat.getColor(requireContext(), android.R.color.white));
17
18        recyclerView = v.findViewById(R.id.viewExplore);
19        btnSearch = v.findViewById(R.id.SearchBar1);
20
21        adapterViewExplore = new AdapterViewExplore(getActivity(), exploreData);
22        recyclerView.setLayoutManager(new GridLayoutManager(getActivity(), 2));
23        recyclerView.setAdapter(adapterViewExplore);
24
25        String urlReq = "http://192.168.100.4/ezgo/router.php";
26
27        RequestQueue queue = Volley.newRequestQueue(requireActivity());
28        Gson gson = new Gson();
29
30        Map<String, Object> params = new HashMap<>();
31        params.put("controller", "location");
32        params.put("method", "explore");
33
34        StringRequest stringRequest = new StringRequest(Request.Method.POST, urlReq, new Response.Listener<String>() {
35            @Override
36            public void onResponse(String response) {
37                Gson gson = new Gson();
38                ExploreData[] exploreData = gson.fromJson(response, ExploreData[].class);
39                adapterViewExplore.setExploreData(exploreData);
40            }
41        }, new Response.ErrorListener() {
42            @Override
43            public void onErrorResponse(VolleyError error) {
44            }
45        });
46
47        queue.add(stringRequest);
48    }
49
50
51
52
53
54
55
56
57
58
59

```

```

ExploreFragment.java
61     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
62         new JSONObject(params),
63         new Response.Listener<JSONObject>() {
64             @Override
65             public void onResponse(JSONObject response) {
66                 ResponseOneObjectList<location> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObjectList<location>>() {}.getType());
67                 boolean success = resp.isSuccess();
68                 List<location> locs = resp.getData();
69
70                 if (success) {
71                     try {
72                         exploreData.clear();
73                         for (location loc : locs) {
74                             exploreData.add(new ExploreData(loc));
75                         }
76                         // Set data ke adapter
77                         adapterViewExplore.setExploreDataList(exploreData);
78                     } catch (Exception e) {
79                         e.printStackTrace();
80                     }
81                 }
82             }
83         },
84         new Response.ErrorListener() {
85             @Override
86             public void onErrorResponse(VolleyError error) { error.printStackTrace(); }
87         });
88     queue.add(jsonObjectRequest);
89
90     btnSearch.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
91         @Override
92
93         public boolean onQueryTextSubmit(String query) { return false; }
94
95         @Override
96         public boolean onQueryTextChange(String newText) {
97             adapterViewExplore.getFilter().filter(newText);
98             return true;
99         }
100     });
101
102     return v;
103 }
104
105 }
106
107 }

```

Kode java ExploreFragment di atas merupakan implementasi dari *ExploreFragment* dalam aplikasi Android yang digunakan untuk menampilkan daftar lokasi eksplorasi dalam tampilan grid dan menyediakan fitur pencarian. Dalam metode *onCreateView*, fragment ini menginisialisasi elemen UI seperti *RecyclerView* dan *SearchView*. Data lokasi diambil dari server menggunakan *Volley* dan dikonversi dari JSON menjadi objek *location* menggunakan *Gson*. Data lokasi kemudian dimasukkan ke dalam *AdapterViewExplore* yang ditampilkan dalam *RecyclerView* dengan tata letak grid. *SearchView* digunakan untuk memfilter hasil pencarian secara real-time. Kode ini mengimplementasikan fitur untuk menampilkan dan memfilter daftar

lokasi eksplorasi, memungkinkan pengguna untuk menjelajahi dan mencari lokasi dengan mudah.

### 3.2.14. HomeFragment

```

 ⑥ HomeFragment.java ✘
1  package com.example.project;
2
3  import ...
4
5
6  public class HomeFragment extends Fragment {
7
8      RecyclerView recyclerView,recyclerView2;
9      AdapterHome adapterHome;
10
11      ArrayList<MyItem> itemList1 = new ArrayList<>();
12      ArrayList<MyItem> itemList2 = new ArrayList<>();
13
14      LinearLayout btnTicket, btnHotel, btnTour, btnSearch;
15      ShapeableImageView btnProfile;
16      TextView uName;
17      String image;
18
19      @Nullable
20      @Override
21      public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
22          View v = inflater.inflate(R.layout.activity_home,container, attachToRoot: false);
23          SharedPreferences preferences = getActivity().getSharedPreferences( name: "ezgo", Context.MODE_PRIVATE);
24          String userJson = preferences.getString( s: "user", s1: null);
25          User user = new Gson().fromJson(userJson, User.class);
26          Log.d( tag: "Ezgo", msg: "uName2: " + user.uName);
27
28          uName = v.findViewById(R.id.profileName);
29          btnSearch = v.findViewById(R.id.search);
30          btnTicket = v.findViewById(R.id.btnTicket);
31          btnHotel = v.findViewById(R.id.btnHotel);
32          btnTour = v.findViewById(R.id.btnTour);
33
34          btnProfile = v.findViewById(R.id.profileHome);
35
36          String urlReq = "http://192.168.100.4/ezgo/router.php";
37          String urlImg = "http://192.168.100.4/ezgo/images/";
38
39          uName.setText(user.uName);
40          if(user.uProfilePicture != null){
41              image = urlImg + user.uProfilePicture;
42          }else{
43              image = urlImg + "defaultpfp.png";
44          }
45          Picasso.get().load(image).into(btnProfile);
46
47          RequestQueue queue = Volley.newRequestQueue(requireActivity());
48          Gson gson = new Gson();
49
50          Map<String, Object> params = new HashMap<>();
51          params.put( k: "controller", v: "location");
52          params.put( k: "method", v: "homePopular");
53
54          JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
55              new JSONObject(params),
56              new Response.Listener<JSONObject>() {
57                  @Override
58                  public void onResponse(JSONObject response) {
59                      Log.d( tag: "Ezgo", msg: "ResponseHome: " + response);
60                      ResponseOneObjectList<location> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObjectList<location>>() {}.getType());
61                      boolean success = resp.isSuccess();
62                      List<location> locs = resp.getData();
63
64                      if (success == true) {
65
66
67
68
69
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

```
© HomeFragment.java x
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120 ①↑
121
122
123
124
125
126
127
128
129
130

    try {
        itemList1.clear();
        int num = 1;
        for (location loc : locs) {
            itemList1.add(new MyItem(loc));
            Log.d( tag: "Ezgo" , msg: "currently: " + num);
            num++;
        }
        recyclerView = v.findViewById(R.id.ViewHome1);
        adapterHome = new AdapterHome(getActivity(), itemList1);
        recyclerView.setLayoutManager(new GridLayoutManager(getActivity(), spanCount: 2));
        recyclerView.setAdapter(adapterHome);
    }catch (Exception e){
        Log.e( tag: "Ezgo" , msg: "Error: " + e);
    }
}

},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Log.e( tag: "Ezgo" , msg: "Error: " + error.toString());
        String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
        Log.e( tag: "Ezgo" , msg: "Response: " + responseBody);
    }
);
queue.add(jsonObjectRequest);
```

```
© HomeFragment.java x
131 params.put( k: "method" , v: "homePopular");
132
133 jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
134     new JSONObject(params),
135     new Response.Listener<JSONObject>() {
136
137 ①↑
138     @Override
139     public void onResponse(JSONObject response) {
140         Log.d( tag: "Ezgo" , msg: "ResponseHome: " + response);
141         ResponseThreeObjectList<ticket, hotel, tour> resp = gson.fromJson(response.toString(), new TypeToken<ResponseThreeObjectList<ticket, hotel, tour>() {}.getType());
142         boolean success = resp.isSuccess();
143         List<ticket> ticks = resp.getData1();
144         List<hotel> hotels = resp.getData2();
145         List<tour> tours = resp.getData3();
146
147         if (success == true) {
148             try {
149                 itemList2.clear();
150                 int num = 1;
151                 for (ticket tick : ticks) {
152                     itemList2.add(new MyItem(tick));
153                     Log.d( tag: "Ezgo" , msg: "currently: " + num);
154                     num++;
155                 }
156                 for (hotel hotl : hotels) {
157                     itemList2.add(new MyItem(hotl));
158                     Log.d( tag: "Ezgo" , msg: "currently: " + num);
159                     num++;
160                 }
161                 for (tour tour : tours) {
162                     itemList2.add(new MyItem(tour));
163                     Log.d( tag: "Ezgo" , msg: "currently: " + num);
164                 }
165             } catch (Exception e) {
166                 Log.e( tag: "Ezgo" , msg: "Error: " + e);
167             }
168         }
169     }
170 }
```

```

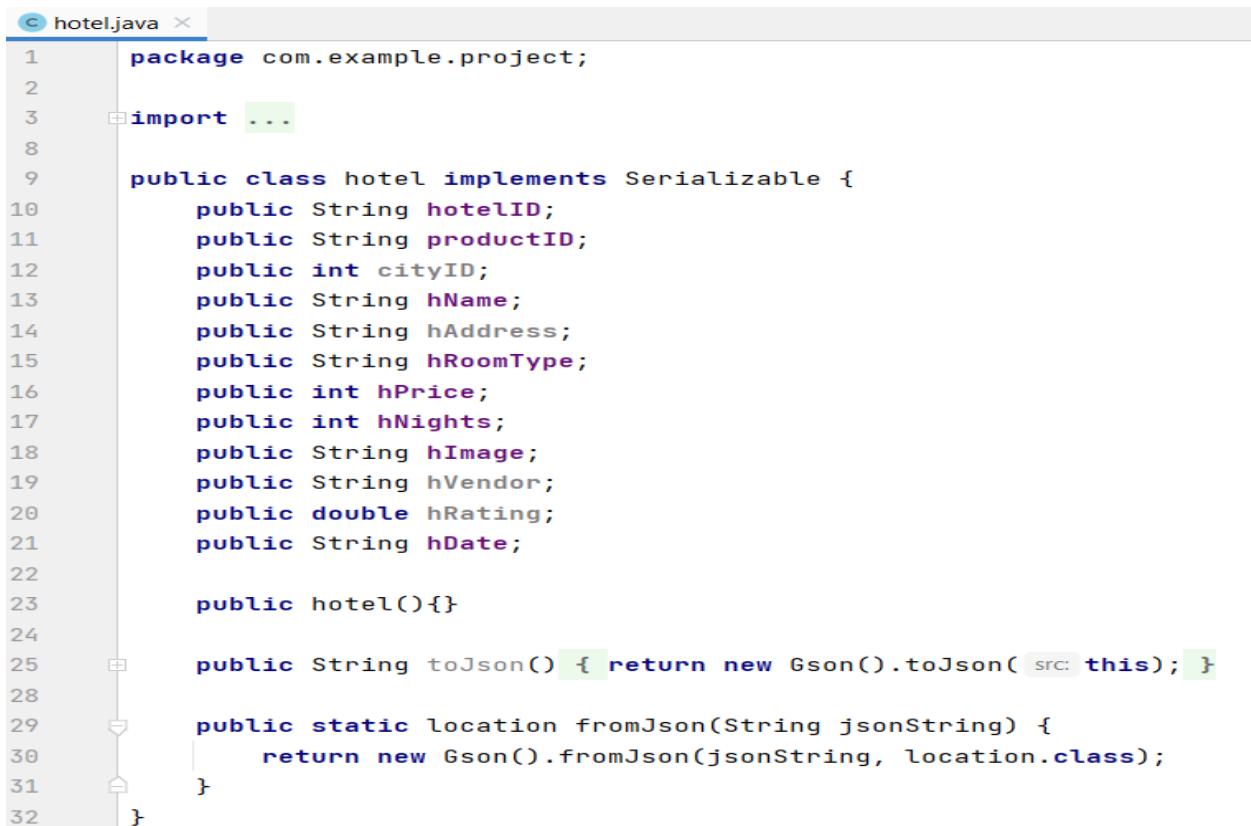
  C HomeFragment.java x
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176 ①↑
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
  C HomeFragment.java x
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211

```

Kode java HomeFragment di atas merupakan implementasi dari *HomeFragment* dalam aplikasi Android yang digunakan untuk menampilkan halaman utama dengan daftar lokasi populer dan produk populer (tiket, hotel, tour). Dalam metode *onCreateView*, fragment ini menginisialisasi elemen UI seperti *RecyclerView*, *LinearLayout*, dan *ShapeableImageView*, serta

mengambil data pengguna dari *SharedPreferences*. Data lokasi populer dan produk populer diambil dari server menggunakan *Volley* dan dikonversi dari JSON menjadi objek Java menggunakan *Gson*. Data tersebut kemudian ditampilkan dalam dua *RecyclerView* menggunakan adapter *AdapterHome*. Terdapat juga pengaturan klik untuk navigasi ke berbagai aktivitas lain seperti *TicketActivity*, *HotelActivity*, *TourActivity*, dan *SearchActivity*, serta fragment profil. Kode ini mengimplementasikan fitur untuk menampilkan halaman beranda yang interaktif dan informatif dengan data dinamis dari server, serta menyediakan navigasi yang mudah ke bagian lain dari aplikasi.

### 3.2.15. hotel

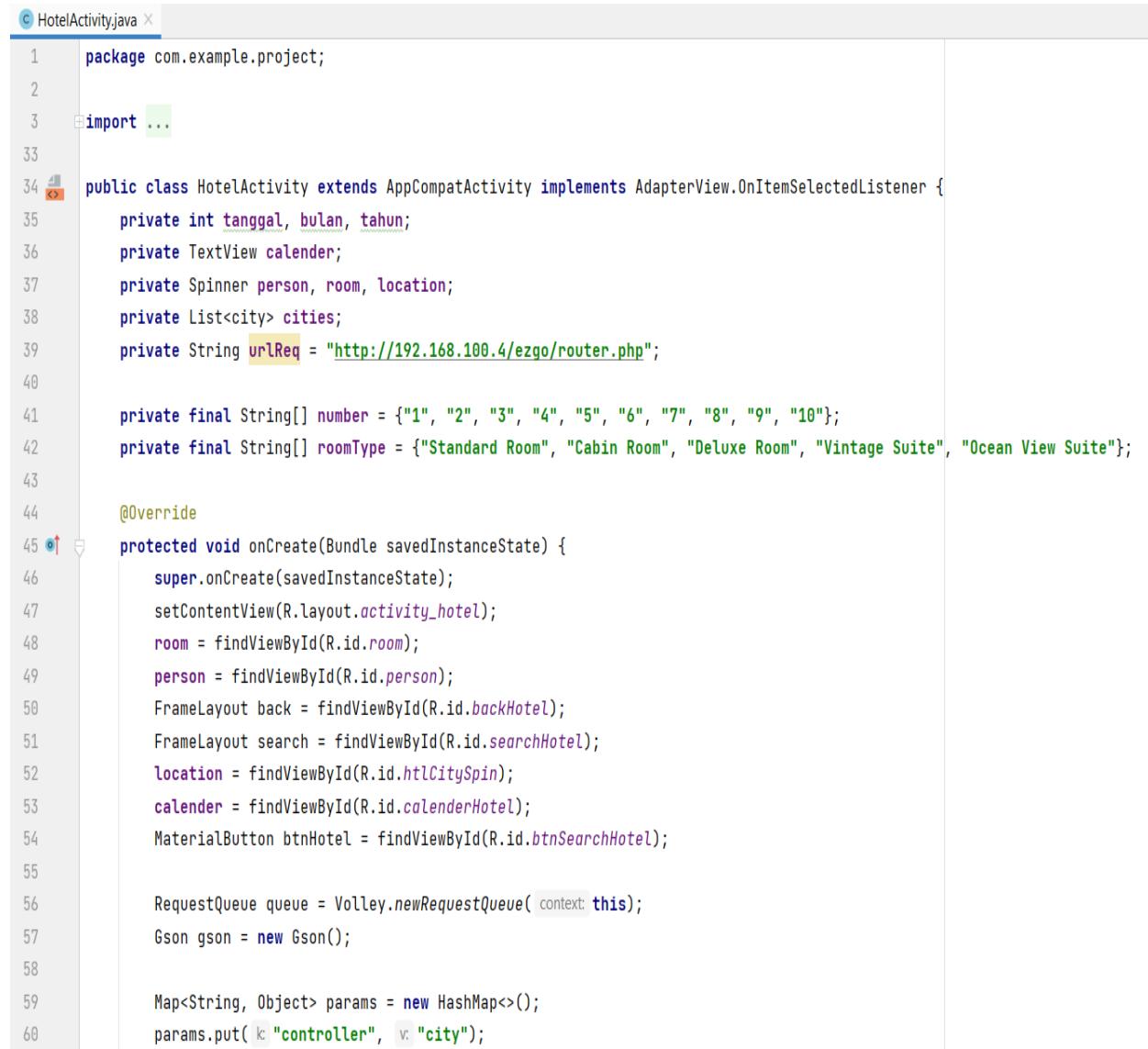


```
1 package com.example.project;
2
3 import ...
4
5 public class hotel implements Serializable {
6     public String hotelID;
7     public String productID;
8     public int cityID;
9     public String hName;
10    public String hAddress;
11    public String hRoomType;
12    public int hPrice;
13    public int hNights;
14    public String hImage;
15    public String hVendor;
16    public double hRating;
17    public String hDate;
18
19    public hotel(){}
20
21    public String toJson() { return new Gson().toJson( src: this); }
22
23    public static location fromJson(String jsonString) {
24        return new Gson().fromJson(jsonString, location.class);
25    }
26}
```

Kode java hotel di atas merupakan implementasi dari kelas *hotel* dalam aplikasi Android yang digunakan untuk merepresentasikan data hotel. Kelas ini mengimplementasikan interface *Serializable* untuk memungkinkan objek hotel diserialisasi. Kelas ini memiliki berbagai atribut seperti *hotelID*, *productID*, *cityID*, *hName*, *hAddress*, *hRoomType*, *hPrice*, *hNights*, *hImage*, *hVendor*, *hRating*, dan *hDate* yang merepresentasikan informasi terkait hotel. Kelas ini juga menyediakan dua metode: *toJson()* untuk mengonversi objek *hotel* menjadi representasi JSON

menggunakan Gson, dan *fromJson()* untuk mengonversi string JSON kembali menjadi objek *hotel*. Kode ini mengimplementasikan fitur serialisasi dan deserialisasi data hotel, yang berguna untuk menyimpan dan mengirimkan data hotel dalam format JSON.

### 3.2.16. HotelActivity



```
1 package com.example.project;
2
3 import ...
33
34 public class HotelActivity extends AppCompatActivity implements AdapterView.OnItemClickListener {
35     private int tanggal, bulan, tahun;
36     private TextView calender;
37     private Spinner person, room, location;
38     private List<city> cities;
39     private String urlReq = "http://192.168.100.4/ezgo/router.php";
40
41     private final String[] number = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10"};
42     private final String[] roomType = {"Standard Room", "Cabin Room", "Deluxe Room", "Vintage Suite", "Ocean View Suite"};
43
44     @Override
45     protected void onCreate(Bundle savedInstanceState) {
46         super.onCreate(savedInstanceState);
47         setContentView(R.layout.activity_hotel);
48         room = findViewById(R.id.room);
49         person = findViewById(R.id.person);
50         FrameLayout back = findViewById(R.id.backHotel);
51         FrameLayout search = findViewById(R.id.searchHotel);
52         location = findViewById(R.id.hotelCitySpin);
53         calender = findViewById(R.id.calenderHotel);
54         MaterialButton btnHotel = findViewById(R.id.btnSearchHotel);
55
56         RequestQueue queue = Volley.newRequestQueue(context: this);
57         Gson gson = new Gson();
58
59         Map<String, Object> params = new HashMap<>();
60         params.put(k: "controller", v: "city");
```

```
© HotelActivity.java x
63     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
64         new JSONObject(params),
65         response -> {
66             Log.d( tag: "Ezgo", msg: "ResponseHome: " + response);
67             ResponseOneObjectList<city> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObjectList<city>>() {
68                 .getType());
69             boolean success = resp.isSuccess();
70             cities = resp.getData();
71
72             if (success == true) {
73                 try {
74                     List<String> cityNames = new ArrayList<>();
75
76                     for (city city : cities) {
77                         cityNames.add(city.cName);
78                         Log.d( tag: "Ezgo", msg: "ResponseHome: " + cityNames);
79                     }
80
81                     ArrayAdapter<String> adapterFrom = new ArrayAdapter<>( context: HotelActivity.this, android.R.layout.simple_spinner_item, cityNames);
82                     location.setAdapter(adapterFrom);
83                     location.setOnItemSelectedListener(HotelActivity.this);
84                 } catch (Exception e) {
85                     Log.e( tag: "Ezgo", msg: "Error: " + e);
86                 }
87             },
88             error -> {
89                 Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
90                 String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
91                 Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
92             });
93     });
© HotelActivity.java x
94     queue.add(jsonObjectRequest);
95
96     ArrayAdapter<String> adapterRoom = new ArrayAdapter<>( context: this, android.R.layout.simple_spinner_item, roomType);
97     room.setAdapter(adapterRoom);
98     room.setOnItemSelectedListener(this);
99
100
101     ArrayAdapter<String> adapterPerson = new ArrayAdapter<>( context: this, android.R.layout.simple_spinner_item, number);
102     person.setAdapter(adapterPerson);
103     person.setOnItemSelectedListener(this);
104
105     calender.setOnClickListener(view -> {
106         Calendar calendar = Calendar.getInstance();
107         tahun = calendar.get(Calendar.YEAR);
108         bulan = calendar.get(Calendar.MONTH);
109         tanggal = calendar.get(Calendar.DAY_OF_MONTH);
110
111         DatePickerDialog dialog = new DatePickerDialog( context: HotelActivity.this, (datePicker, i, i1, i2) -> {
112             tahun = i;
113             bulan = i1 + 1;
114             tanggal = i2;
115
116             calender.setText(String.format("%04d-%02d-%02d", tahun, bulan, tanggal));
117             }, tahun, bulan, tanggal);
118             dialog.show();
119     });
120
121     btnHotel.setOnClickListener(view -> {
122         int City = 0;
```

```

  c HotelActivity.java x
125     String targetCity = location.getSelectedItem().toString();
126
127     for (city obj : cities) {
128         if (targetCity.equals(obj.cName)) {
129             City = obj.cityID;
130         }
131     }
132
133     String date = calender.getText().toString();
134     int per = Integer.parseInt(person.getSelectedItem().toString());
135     String rooms = room.getSelectedItem().toString();
136
137     Intent i = new Intent(getApplicationContext(), HotelViewActivity.class);
138     i.putExtra( name: "loc", City);
139     i.putExtra( name: "date", date);
140     i.putExtra( name: "room", rooms);
141     i.putExtra( name: "per", per);
142     startActivity(i);
143 );
144
145     back.setOnClickListener(view -> onBackPressed());
146
147     search.setOnClickListener(view -> {
148         Intent i = new Intent(getApplicationContext(), SearchActivity.class);
149         startActivity(i);
150     });
151 }

```

```

  c HotelActivity.java x
153     @Override
154     public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
155
156     }
157
158     @Override
159     public void onNothingSelected(AdapterView<?> adapterView) {
160
161     }
162
163     @Override
164     public void onPointerCaptureChanged(boolean hasCapture) {
165         super.onPointerCaptureChanged(hasCapture);
166     }
167 }

```

Kode java HotelActivity di atas merupakan implementasi dari *HotelActivity* dalam aplikasi Android yang digunakan untuk mencari dan menampilkan informasi hotel berdasarkan preferensi pengguna. Aktivitas ini menginisialisasi beberapa elemen UI seperti *Spinner* untuk memilih jumlah orang, jenis kamar, dan lokasi kota, serta *TextView* untuk memilih tanggal. Dengan menggunakan *Volley* untuk melakukan permintaan HTTP POST ke server, aplikasi mengambil daftar kota yang tersedia dan menampilkannya dalam *Spinner*. Ketika pengguna

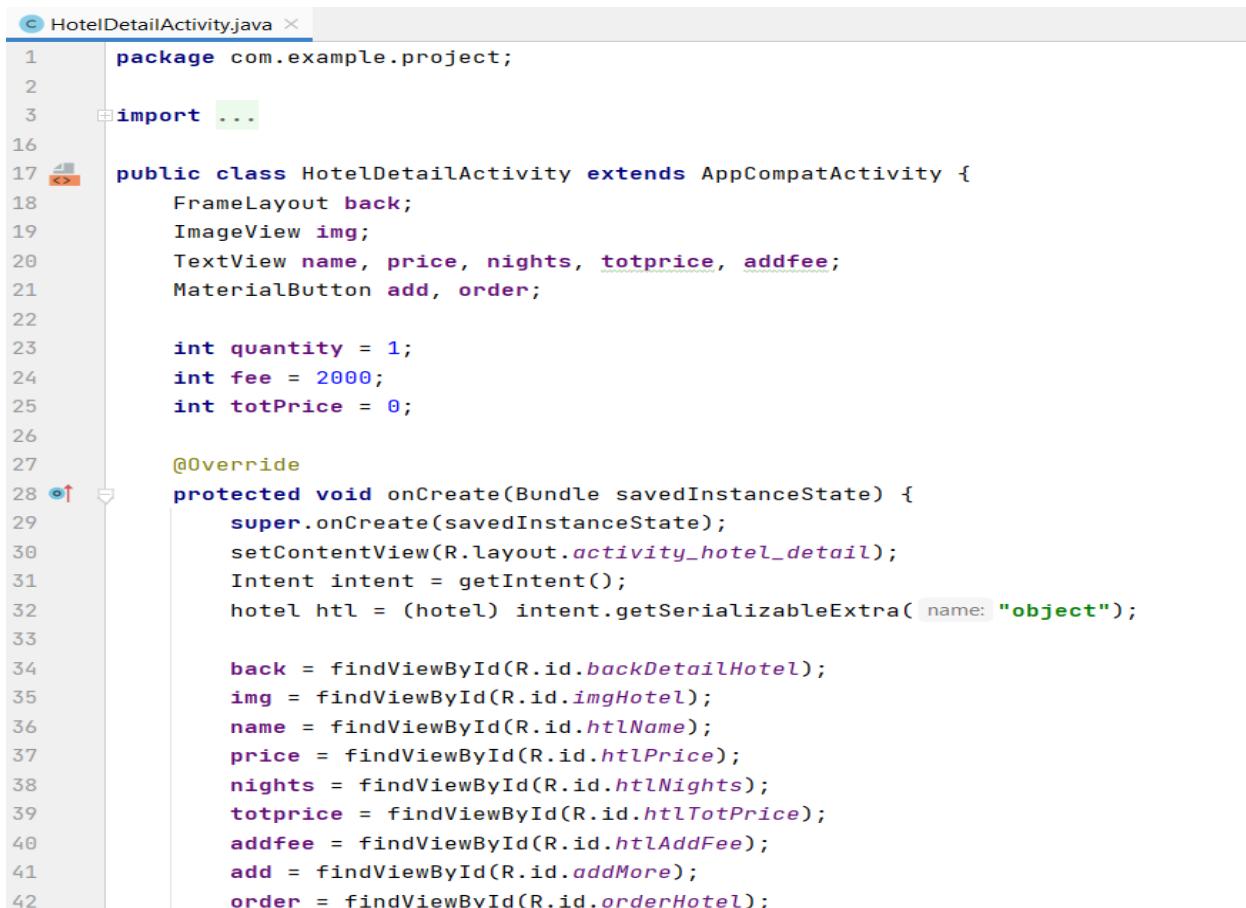
memilih tanggal, jumlah orang, jenis kamar, dan lokasi kota, serta menekan tombol cari, data yang dipilih akan dikirim ke *HotelViewActivity* menggunakan *Intent* untuk menampilkan hasil pencarian. Aktivitas ini juga memiliki fitur untuk kembali ke layar sebelumnya dan navigasi ke halaman pencarian lainnya.

### **3.2.17. HotelData**

```
c HotelData.java x
1  package com.example.project;
2
3  public class HotelData {
4
5      private String Id;
6      private String Name;
7      private String Address;
8      private String RoomType;
9      private int Price;
10     private String Date;
11
12
13     public HotelData(String id, String name, String address, String roomType, int price, String date) {
14         Id = id;
15         Name = name;
16         Address = address;
17         RoomType = roomType;
18         Price = price;
19         Date = date;
20     }
21
22     public String getId() { return Id; }
23
24     public void setId(String id) { Id = id; }
25
26     public String getName() { return Name; }
27
28     public void setName(String name) { Name = name; }
29
30     public String getAddress() { return Address; }
31     public void setAddress(String address) { Address = address; }
32
33     public String getRoomType() { return RoomType; }
34
35     public void setRoomType(String roomType) { RoomType = roomType; }
36
37     public int getPrice() { return Price; }
38
39     public void setPrice(int price) { Price = price; }
40
41     public String getDate() { return Date; }
42
43     public void setDate(String date) { Date = date; }
44 }
```

Kode java HotelData di atas merupakan implementasi dari kelas *HotelData* dalam aplikasi Android yang digunakan untuk merepresentasikan data hotel secara sederhana. Kelas ini memiliki beberapa atribut seperti *Id*, *Name*, *Address*, *RoomType*, *Price*, dan *Date* yang merepresentasikan informasi dasar terkait hotel. Kelas ini menyediakan konstruktor untuk menginisialisasi objek *HotelData* dan metode *getter* serta *setter* untuk mengakses dan memodifikasi nilai dari atribut-atribut tersebut. Kode ini mengimplementasikan fitur encapsulation dalam OOP untuk mengelola data hotel dengan menyediakan akses terkontrol ke properti hotel, yang penting untuk manipulasi data di berbagai bagian aplikasi.

### 3.2.18. HotelDetailActivity



```
1 package com.example.project;
2
3 import ...
4
5
6 public class HotelDetailActivity extends AppCompatActivity {
7     FrameLayout back;
8     ImageView img;
9     TextView name, price, nights, totprice, addfee;
10    MaterialButton add, order;
11
12    int quantity = 1;
13    int fee = 2000;
14    int totPrice = 0;
15
16
17    @Override
18    protected void onCreate(Bundle savedInstanceState) {
19        super.onCreate(savedInstanceState);
20        setContentView(R.layout.activity_hotel_detail);
21        Intent intent = getIntent();
22        hotel htl = (hotel) intent.getSerializableExtra("object");
23
24        back = findViewById(R.id.backDetailHotel);
25        img = findViewById(R.id.imgHotel);
26        name = findViewById(R.id.htlName);
27        price = findViewById(R.id.htlPrice);
28        nights = findViewById(R.id.htlNights);
29        totprice = findViewById(R.id.htlTotPrice);
30        addfee = findViewById(R.id.htlAddFee);
31        add = findViewById(R.id.addMore);
32        order = findViewById(R.id.orderHotel);
33
34
35
36
37
38
39
40
41
42
```

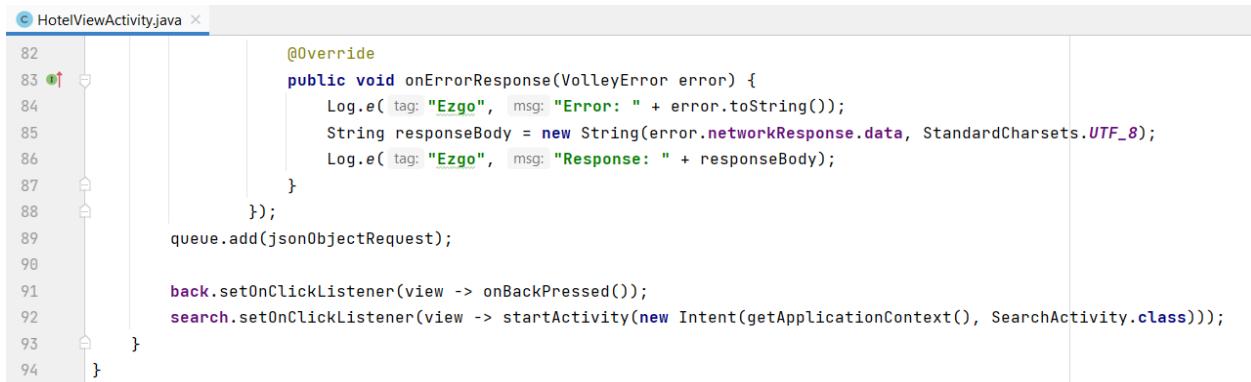
```
HotelDetailActivity.java
44     DecimalFormat decimalFormat = new DecimalFormat( pattern: "#,###");
45
46     totPrice = htl.hPrice * quantity + fee;
47
48     String urlImg = "http://192.168.100.4/ezgo/images/";
49     String image = urlImg + htl.hImage;
50
51     Picasso.get().load(image).into(img);
52
53     name.setText(htl.hName);
54     price.setText(decimalFormat.format(htl.hPrice));
55     nights.setText(Integer.toString(quantity));
56     addfee.setText(decimalFormat.format(fee));
57     totprice.setText(decimalFormat.format(totPrice));
58
59     back.setOnClickListener(view -> onBackPressed());
60
61     add.setOnClickListener(view -> {
62         if((quantity + 1) <= htl.hNights){
63             quantity++;
64             totPrice = htl.hPrice * quantity + fee;
65             nights.setText(Integer.toString(quantity));
66             totprice.setText(decimalFormat.format(totPrice));
67         }else{
68             Toast.makeText(getApplicationContext(), text: "Max Amount Reached", Toast.LENGTH_LONG).show();
69         }
70     });
71
72     order.setOnClickListener(view -> {
73         Intent i = new Intent(getApplicationContext(), PaymentActivity.class);
74         i.putExtra( name: "object", htl);
75         i.putExtra( name: "price", totPrice);
76         i.putExtra( name: "amount", quantity);
77         startActivity(i);
78     });
79 }
80 }
```

Kode java HotelDetailActivity di atas merupakan implementasi dari *HotelDetailActivity* dalam aplikasi Android yang digunakan untuk menampilkan detail informasi sebuah hotel dan memungkinkan pengguna untuk melakukan pemesanan. Aktivitas ini menerima objek *hotel* dari *Intent*, kemudian menampilkan informasi hotel seperti nama, harga, jumlah malam, dan biaya tambahan. Gambar hotel ditampilkan menggunakan pustaka *Picasso*. Pengguna dapat menambah jumlah malam menginap dengan tombol “*add*”, yang secara otomatis menghitung dan menampilkan total harga. Tombol “*order*” memungkinkan pengguna untuk melanjutkan ke halaman pembayaran dengan mengirimkan detail pemesanan melalui *Intent*. Kode ini mengimplementasikan fitur tampilan detail hotel, penghitungan harga dinamis, dan navigasi ke proses pembayaran.

### 3.2.19. HotelViewActivity

```
1  package com.example.project;
2
3  import ...
4
5
6  public class HotelViewActivity extends AppCompatActivity {
7      RecyclerView recyclerView;
8      AdapterViewHotel adapterViewHotel;
9      FrameLayout back, search;
10     String urlReq = "http://192.168.100.4/ezgo/router.php";
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_hotel_view);
16         recyclerView = findViewById(R.id.viewHotel);
17         back = findViewById(R.id.backHotelView);
18         search = findViewById(R.id.searchHotelView);
19
20         Intent intent = getIntent();
21         int cityText = intent.getIntExtra("loc", 0);
22         String dateText = intent.getStringExtra("date");
23         int personText = intent.getIntExtra("per", 0);
24         String roomText = intent.getStringExtra("room");
25
26         RequestQueue queue = Volley.newRequestQueue(context: this);
27         Gson gson = new Gson();
28
29         Map<String, Object> params = new HashMap<>();
30         params.put("controller", "order");
31         params.put("method", "searchHotel");
32         params.put("cityID", cityText);
33
34     }
35
36     @Override
37     protected void onStart() {
38         super.onStart();
39         adapterViewHotel = new AdapterViewHotel(this, recyclerView, params);
40         recyclerView.setAdapter(adapterViewHotel);
41     }
42
43     @Override
44     protected void onStop() {
45         super.onStop();
46         adapterViewHotel.onDestroy();
47     }
48
49     @Override
50     protected void onDestroy() {
51         super.onDestroy();
52         adapterViewHotel.onDestroy();
53     }
54
55 }
```

```
56     params.put( k "hRoomType", roomText);
57     params.put( k "hDate", dateText);
58     params.put( k "hNights", personText);
59
60     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
61         new JSONObject(params),
62         new Response.Listener<JSONObject>() {
63             @Override
64             public void onResponse(JSONObject response) {
65                 Log.d( tag: "Ezgo", msg: "ResponseHome: " + response);
66                 ResponseOneObjectList<hotel> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObjectList<hotel>>() {}.getType());
67                 boolean success = resp.isSuccess();
68                 List<hotel> hotelData = resp.getData();
69
70                 if (success == true) {
71                     try {
72                         adapterViewHotel = new AdapterViewHotel( context: HotelViewActivity.this, hotelData);
73                         recyclerView.setLayoutManager(new LinearLayoutManager( context: HotelViewActivity.this));
74                         recyclerView.setAdapter(adapterViewHotel);
75                     }catch (Exception e){
76                         Log.e( tag: "Ezgo", msg: "Error: " + e);
77                     }
78                 }
79             }
80         },
81         new Response.ErrorListener() {
```



```
82
83     @Override
84     public void onTaskComplete(JSONObject response) {
85         Log.e( tag: "Ezgo", msg: "Task Complete: " + response.toString());
86         String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
87         Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
88     });
89     queue.add(jsonObjectRequest);
90 }
91 back.setOnClickListener(view -> onBackPressed());
92 search.setOnClickListener(view -> startActivityForResult(new Intent(getApplicationContext(), SearchActivity.class)));
93 }
94 }
```

Kode java HotelViewActivity di atas merupakan implementasi dari *HotelViewActivity* dalam aplikasi Android yang digunakan untuk menampilkan daftar hotel berdasarkan kriteria pencarian pengguna. Aktivitas ini menginisialisasi elemen UI seperti *RecyclerView* untuk menampilkan daftar hotel, serta tombol untuk kembali dan melakukan pencarian ulang. Data pencarian seperti ID kota, tanggal, jumlah orang, dan jenis kamar diterima melalui *Intent*. Dengan menggunakan *Volley* untuk mengirim permintaan HTTP POST ke server, data hotel yang sesuai dengan kriteria tersebut diambil dan diubah menjadi objek *hotel* menggunakan *Gson*. Jika permintaan berhasil, data hotel akan ditampilkan dalam *RecyclerView* menggunakan *AdapterViewHotel*. Aktivitas ini juga mengimplementasikan fitur navigasi ke halaman pencarian ulang dan kembali ke halaman sebelumnya. Kode ini mengimplementasikan pencarian dinamis dan tampilan daftar hotel berdasarkan preferensi pengguna.

### 3.2.20. internalDB



```
1 package com.example.project;
2
3 import ...
4
5 public class internalDB extends SQLiteOpenHelper {
6     private static final String DATABASE_NAME = "internal.db";
7     private static final int DATABASE_VERSION = 1;
8
9     private SQLiteDatabase db;
10    private String[] allColumns =
11        {"userID", "uName", "uAddress", "uPhone", "uEmail", "uBirthdate", "uProfilePicture"};
12
13    public internalDB(Context context) {
14        super(context, DATABASE_NAME, null, DATABASE_VERSION);
15        db = getWritableDatabase();
16    }
17 }
```

```
internalDB.java x
26
27 ① @Override
28  public void onCreate(SQLiteDatabase db) {
29      String createTableQuery = "CREATE TABLE IF NOT EXISTS user (\n" +
30          "    userID TEXT PRIMARY KEY,\n" +
31          "    uName TEXT NOT NULL,\n" +
32          "    uAddress TEXT,\n" +
33          "    uPhone TEXT,\n" +
34          "    uEmail TEXT,\n" +
35          "    uBirthdate TEXT,\n" +
36          "    uProfilePicture TEXT\n" +
37      ");";
38      db.execSQL(createTableQuery);
39
40 ① @Override
41  public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
42      sqLiteDatabase.execSQL("DROP TABLE IF EXISTS user");
43      onCreate(sqLiteDatabase);
44
45
46 ① private User cursorToUser(Cursor cursor){
47     User user = new User();
48     user.userID = cursor.getString( 0 );
49     user.uName = cursor.getString( 1 );
50     user.uAddress = cursor.getString( 2 );
51     user.uPhone = cursor.getString( 3 );
52     user.uEmail = cursor.getString( 4 );
53     user.uBirthdate = cursor.getString( 5 );
54     user.uProfilePicture = cursor.getString( 6 );
55
56     return user;
57 }
```

```
internalDB.java x
58 ① public void createUser(User user){
59      ContentValues values = new ContentValues();
60      values.put("userID", user.userID);
61      values.put("uName", user.uName);
62      values.put("uAddress", user.uAddress);
63      values.put("uPhone", user.uPhone);
64      values.put("uEmail", user.uEmail);
65      values.put("uBirthdate", user.uBirthdate != null ? user.uBirthdate.toString() : null);
66      values.put("uProfilePicture", user.uProfilePicture);
67      db.insert( table: "user", nullColumnHack: null, values);
68  }
69
70 ① public User getUser(){
71     User user = new User();
72     Cursor cursor = db.query( table: "user", allColumns, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null );
73     if (cursor != null && cursor.moveToFirst()) {
74         user = cursorToUser(cursor);
75         cursor.close();
76     }
77     return user;
78  }
79
80 ① public void updateUser(User user){
81     String filter = "userID=" + user.userID;
82     ContentValues values = new ContentValues();
83     values.put("userID", user.userID);
84     values.put("uName", user.uName);
85     values.put("uAddress", user.uAddress);
86     values.put("uPhone", user.uPhone);
87     values.put("uEmail", user.uEmail);
88     values.put("uBirthdate", user.uBirthdate.toString());
```

```
internalDB.java
89         values.put("uProfilePicture", user.uProfilePicture);
90         db.update( table: "user", values, filter, whereArgs: null);
91     }
92
93     @
94     public void deleteUser(User user){
95         String filter = "userID=?";
96         db.delete( table: "user", filter, new String[]{user.userID});
97     }
98
99     public boolean checkUserExist() {
100         String query = "SELECT COUNT(*) FROM user";
101         Cursor cursor = db.rawQuery(query, selectionArgs: null);
102         boolean ret = false;
103
104         if (cursor != null && cursor.moveToFirst()) {
105             int count = cursor.getInt( i: 0);
106             ret = count > 0;
107             cursor.close();
108         }
109
110         return ret;
111     }
112
113     @Override
114     public synchronized void close() {
115         if (db != null) {
116             db.close();
117             super.close();
118         }
119     }
}
```

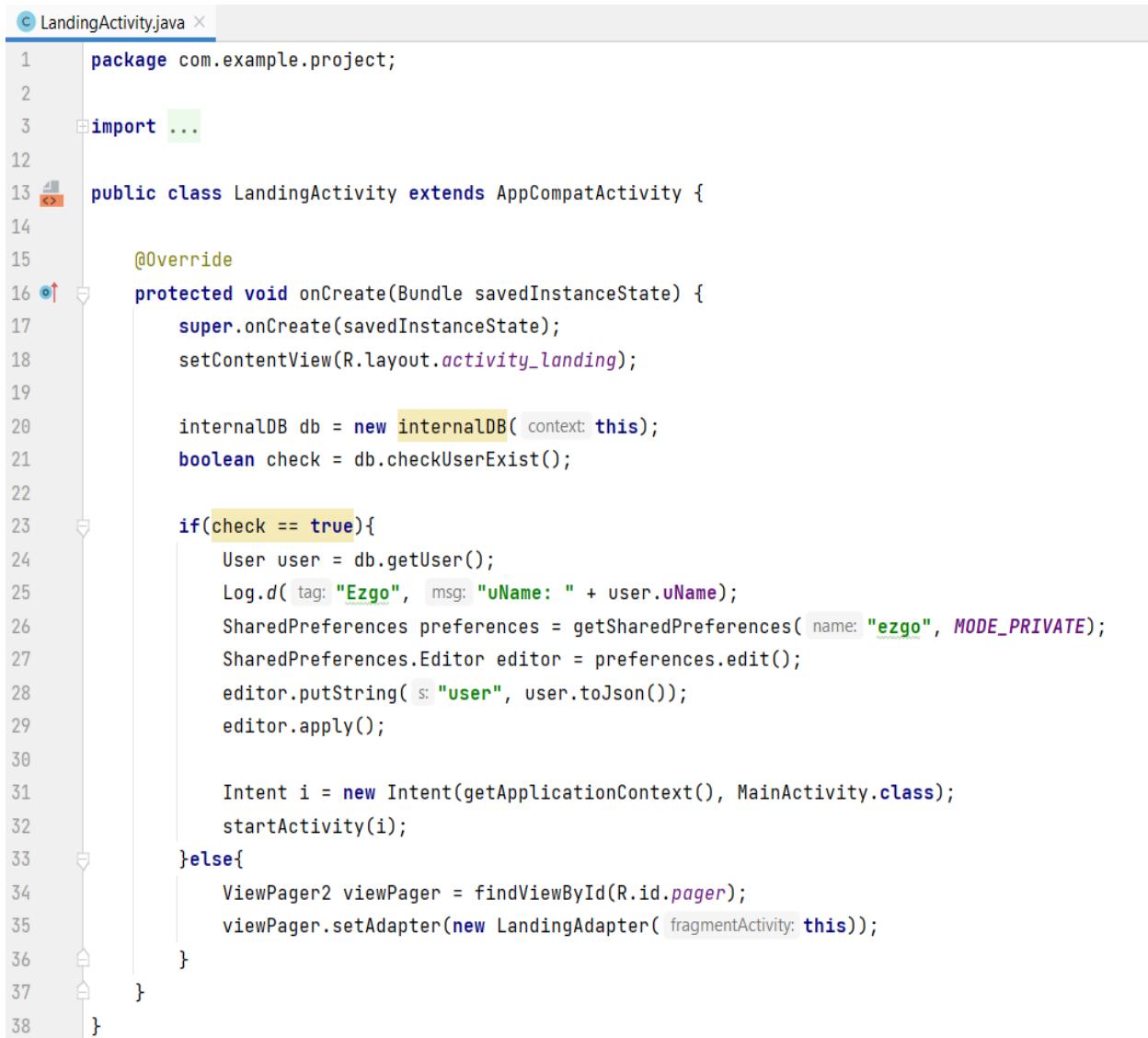
Kode java internalDB di atas merupakan implementasi dari kelas *internalDB* yang berfungsi sebagai *SQLiteOpenHelper* untuk mengelola database SQLite dalam aplikasi Android. Kelas ini bertanggung jawab untuk membuat, memperbarui, dan mengelola tabel *user* yang menyimpan informasi pengguna seperti ID, nama, alamat, telepon, email, tanggal lahir, dan foto profil. Metode *onCreate* digunakan untuk membuat tabel jika belum ada, dan *onUpgrade* untuk memperbarui skema tabel. Kelas ini juga menyediakan metode untuk operasi CRUD (Create, Read, Update, Delete) pada data pengguna, seperti *createUser*, *getUser*, *updateUser*, dan *deleteUser*. Selain itu, ada metode *checkUserExist* untuk memeriksa apakah ada data pengguna di tabel. Implementasi ini mengelola penyimpanan data pengguna secara lokal dan menyediakan antarmuka untuk berinteraksi dengan database SQLite.

### 3.2.21. KotaActivity

```
c KotaActivity.java x
48     } else if (choose.equalsIgnoreCase( anotherString: "Tour")) {
49         Intent resultIntent = new Intent(getApplicationContext(), TourActivity.class);
50         resultIntent.putExtra( name: "kota", selectedItem);
51         startActivity(resultIntent);
52     } else if (choose.equalsIgnoreCase( anotherString: "Hotel")) {
53         Intent resultIntent = new Intent(getApplicationContext(), HotelActivity.class);
54         resultIntent.putExtra( name: "kota", selectedItem);
55         startActivity(resultIntent);
56     }
57     finish();
58 });
59
60     searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
61         @Override
62         public boolean onQueryTextSubmit(String query) { return false; }
63
64         @Override
65         public boolean onQueryTextChange(String newText) {
66             adapter.getFilter().filter(newText);
67             return true;
68         }
69     });
70 }
71 }
72 }
73 }
```

Kode java KotaActivity di atas merupakan implementasi aktivitas Android yang menampilkan daftar kota dalam bentuk *ListView* dan memungkinkan pengguna untuk mencari kota menggunakan *SearchView*. Daftar kota diisi dengan data dummy (“Item0” hingga “Item13”) dan ditampilkan menggunakan *ArrayAdapter*. Saat pengguna memilih kota dari daftar, aktivitas memeriksa jenis pilihan (*choose*) yang diteruskan dari aktivitas sebelumnya untuk menentukan tindakan yang tepat: mengembalikan hasil ke aktivitas pemanggil (untuk “Ticket”), atau memulai *TourActivity* (untuk “Tour”), atau memulai *HotelActivity* (untuk “Hotel”). Fitur pencarian memungkinkan pengguna untuk memfilter daftar kota berdasarkan teks yang dimasukkan di *SearchView*.

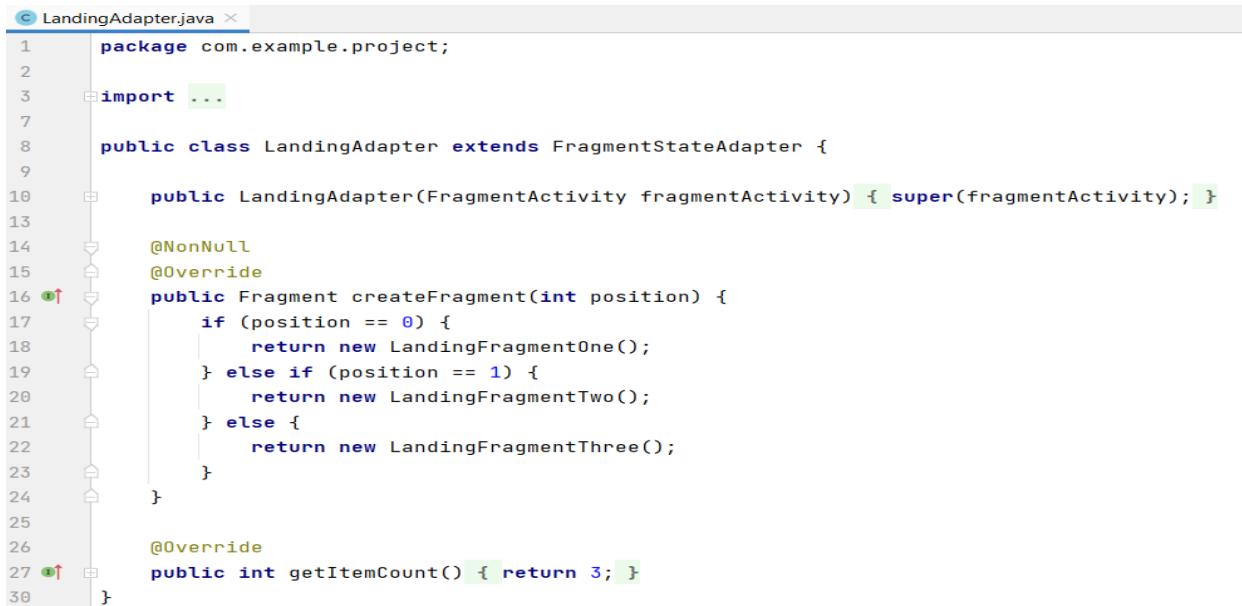
### 3.2.22. LandingActivity



```
1 package com.example.project;
2
3 import ...
4
5
6
7
8
9
10
11
12
13 public class LandingActivity extends AppCompatActivity {
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_landing);
19
20         internalDB db = new internalDB( context: this);
21         boolean check = db.checkUserExist();
22
23         if(check == true){
24             User user = db.getUser();
25             Log.d( tag: "Ezgo", msg: "uName: " + user.uName);
26             SharedPreferences preferences = getSharedPreferences( name: "ezgo", MODE_PRIVATE);
27             SharedPreferences.Editor editor = preferences.edit();
28             editor.putString( s: "user", user.toJson());
29             editor.apply();
30
31             Intent i = new Intent(getApplicationContext(), MainActivity.class);
32             startActivity(i);
33         }else{
34             ViewPager2 viewPager = findViewById(R.id.pager);
35             viewPager.setAdapter(new LandingAdapter( fragmentActivity: this));
36         }
37     }
38 }
```

Kode java LandingActivity di atas merupakan implementasi aktivitas Android yang berfungsi sebagai layar pembuka atau *landing page* untuk aplikasi. Pada *onCreate*, aktivitas ini memeriksa apakah ada pengguna yang tersimpan di database lokal (*internalDB*). Jika pengguna ada, data pengguna diambil dari database dan disimpan ke *SharedPreferences*, kemudian pengguna dialihkan ke *MainActivity*. Jika tidak ada pengguna yang ditemukan, maka *ViewPager2* digunakan untuk menampilkan serangkaian halaman pengantar menggunakan adapter *LandingAdapter*. Fitur ini memungkinkan aplikasi untuk memberikan pengalaman pengantar kepada pengguna baru atau langsung melanjutkan ke halaman utama bagi pengguna yang sudah terdaftar.

### 3.2.23. LandingAdapter



```
1 package com.example.project;
2
3 import ...
4
5
6 public class LandingAdapter extends FragmentStateAdapter {
7
8     public LandingAdapter(FragmentActivity fragmentActivity) { super(fragmentActivity); }
9
10    @NonNull
11    @Override
12    public Fragment createFragment(int position) {
13        if (position == 0) {
14            return new LandingFragmentOne();
15        } else if (position == 1) {
16            return new LandingFragmentTwo();
17        } else {
18            return new LandingFragmentThree();
19        }
20    }
21
22    @Override
23    public int getItemCount() { return 3; }
24
25
26
27
28
29
30 }
```

Kode java LandingAdapter di atas merupakan implementasi adapter untuk *ViewPager2* yang mengelola dan menyediakan tiga fragmen (*LandingFragmentOne*, *LandingFragmentTwo*, dan *LandingFragmentThree*) yang ditampilkan pada halaman pengantar aplikasi. Adapter ini memperluas *FragmentStateAdapter* dan mengimplementasikan metode *createFragment* untuk mengembalikan fragmen yang sesuai berdasarkan posisi halaman yang diminta. Metode *getItemCount* mengembalikan jumlah total fragmen, yaitu tiga. Fitur ini memungkinkan pengguna untuk menavigasi melalui serangkaian layar pengantar dengan menggeser ke kiri atau kanan.

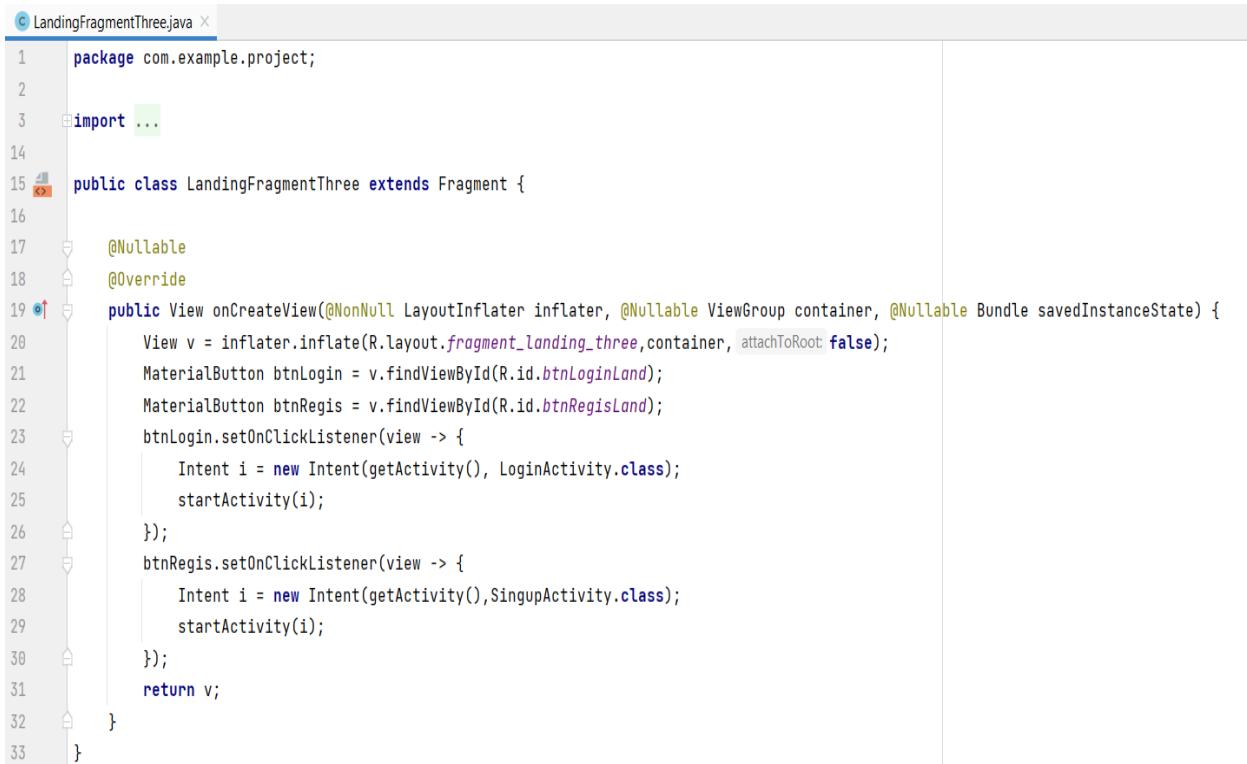
### 3.2.24. LandingFragmentOne



```
1 package com.example.project;
2
3 import ...
4
5
6 public class LandingFragmentOne extends Fragment {
7
8    @Nullable
9    @Override
10    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
11        return inflater.inflate(R.layout.fragment_landing_one, container, false);
12    }
13
14
15
16
17
18
19 }
```

Kode java LandingFragmentOne di atas merupakan implementasi dari sebuah fragmen dalam Android yang menampilkan layout *fragment\_landing\_one*. Kelas ini memperluas *Fragment* dan mengoverride metode *onCreateView* untuk menginisialisasi tampilan fragmen dengan menggunakan *LayoutInflater* untuk menginflate layout XML yang telah ditentukan (*fragment\_landing\_one.xml*). Fungsinya adalah menampilkan konten yang didefinisikan dalam *fragment\_landing\_one.xml* ketika fragmen ini ditampilkan dalam *ViewPager2*. Kode ini adalah bagian dari serangkaian fragmen yang digunakan untuk layar pengantar aplikasi.

### **3.2.25. LandingFragmentThree**



```
1 package com.example.project;
2
3 import ...
4
5
6 public class LandingFragmentThree extends Fragment {
7
8     @Nullable
9     @Override
10    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
11        View v = inflater.inflate(R.layout.fragment_landing_three, container, false);
12        MaterialButton btnLogin = v.findViewById(R.id.btnLoginLand);
13        MaterialButton btnRegis = v.findViewById(R.id.btnRegisLand);
14        btnLogin.setOnClickListener(view -> {
15            Intent i = new Intent(getActivity(), LoginActivity.class);
16            startActivity(i);
17        });
18        btnRegis.setOnClickListener(view -> {
19            Intent i = new Intent(getActivity(), SingupActivity.class);
20            startActivity(i);
21        });
22        return v;
23    }
24
25
26
27
28
29
30
31
32
33 }
```

Kode java LandingFragmentThree di atas merupakan implementasi dari sebuah fragmen yang menampilkan layout *fragment\_landing\_three* dan menyediakan dua tombol interaktif: “Login” dan “Register”. Ketika tombol “Login” diklik, fragmen ini akan memulai aktivitas *LoginActivity*, sedangkan tombol “Register” akan memulai aktivitas *SingupActivity*. Kode ini memungkinkan pengguna untuk memilih untuk masuk atau mendaftar pada aplikasi. Fragmen ini adalah bagian dari layar pengantar aplikasi yang memfasilitasi navigasi ke aktivitas login atau pendaftaran.

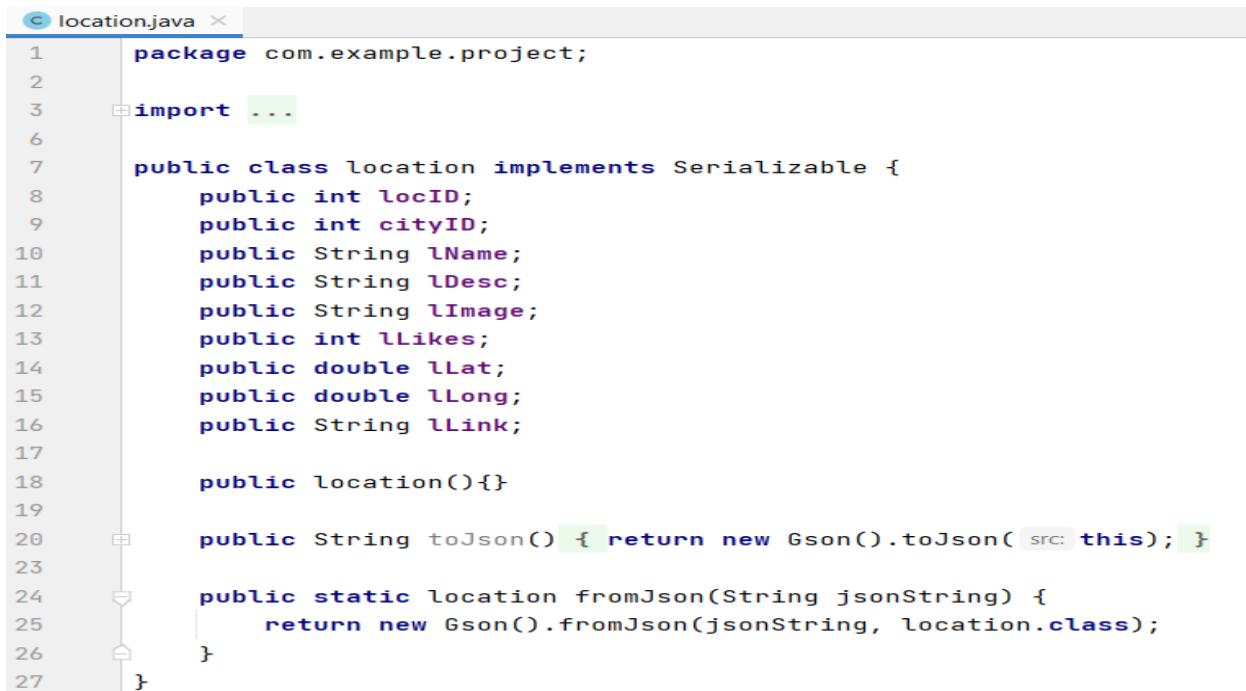
### 3.2.26. LandingFragmentTwo



```
1 package com.example.project;
2
3 import ...
4
5
6
7
8
9
10
11
12 public class LandingFragmentTwo extends Fragment {
13
14     @Nullable
15     @Override
16     public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
17         return inflater.inflate(R.layout.fragment_landing_two, container, attachToRoot: false);
18     }
19 }
```

Kode java LandingFragmentTwo di atas merupakan implementasi dari sebuah fragmen yang mengimplementasikan tampilan dari layout *fragment\_landing\_two*. Fungsi dari fragmen ini adalah untuk menampilkan konten pada layar pengantar aplikasi, dan tampilan ini diatur melalui file layout XML yang terkait. Fragmen ini merupakan bagian dari carousel atau slide pengantar yang ditampilkan kepada pengguna saat pertama kali membuka aplikasi, membantu memandu mereka melalui informasi penting sebelum masuk ke aplikasi.

### 3.2.27. location



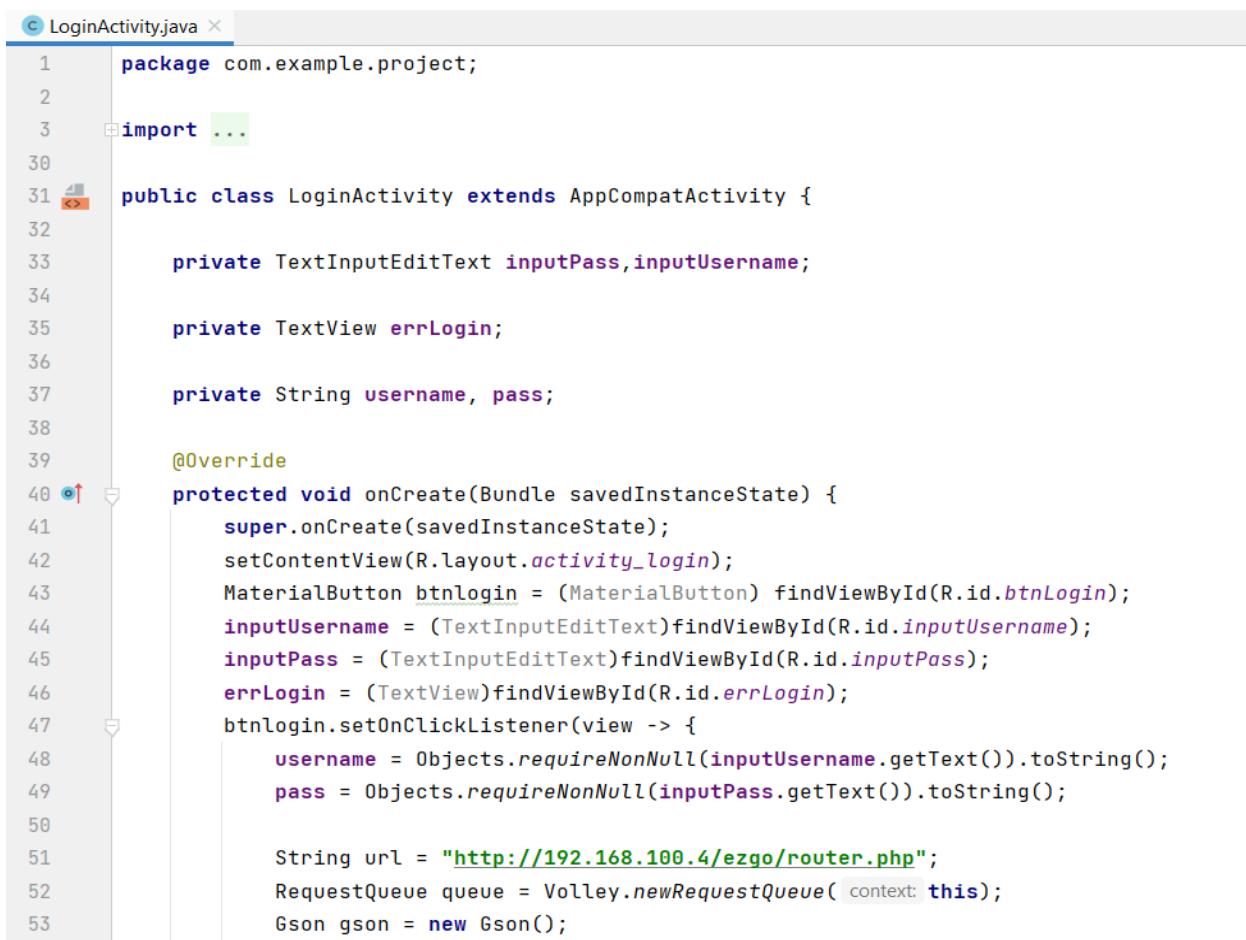
```
1 package com.example.project;
2
3 import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
```

```
public class location implements Serializable {
    public int locID;
    public int cityID;
    public String lName;
    public String lDesc;
    public String lImage;
    public int lLikes;
    public double lLat;
    public double lLong;
    public String lLink;

    public location(){}
    public String toJson() { return new Gson().toJson( src: this); }
    public static location fromJson(String jsonString) {
        return new Gson().fromJson(jsonString, location.class);
    }
}
```

Kode java location di atas merupakan implementasi dari kelas *location* dalam aplikasi Android yang digunakan untuk merepresentasikan objek lokasi. Kelas ini mengimplementasikan antarmuka *Serializable*, sehingga objek *location* dapat dengan mudah diserialisasi dan dideserialisasi. Kelas ini memiliki beberapa atribut: *locID*, *cityID*, *lName* (nama lokasi), *lDesc* (deskripsi lokasi), *lImage* (gambar lokasi), *lLikes* (jumlah suka), *lLat* (latitude), *lLong* (longitude), dan *lLink* (tautan terkait lokasi). Konstruktur default (*location()*) disediakan, serta dua metode: *toJson()* untuk mengonversi objek *location* menjadi string JSON menggunakan Gson, dan *fromJson(String jsonString)* untuk membuat objek *location* dari string JSON. Kode ini mengimplementasikan fitur serialisasi dan deserialisasi JSON yang memungkinkan pertukaran data objek *location* dalam format JSON, yang sangat berguna untuk komunikasi data antar bagian aplikasi atau antara aplikasi dan server.

### 3.2.28. LoginActivity



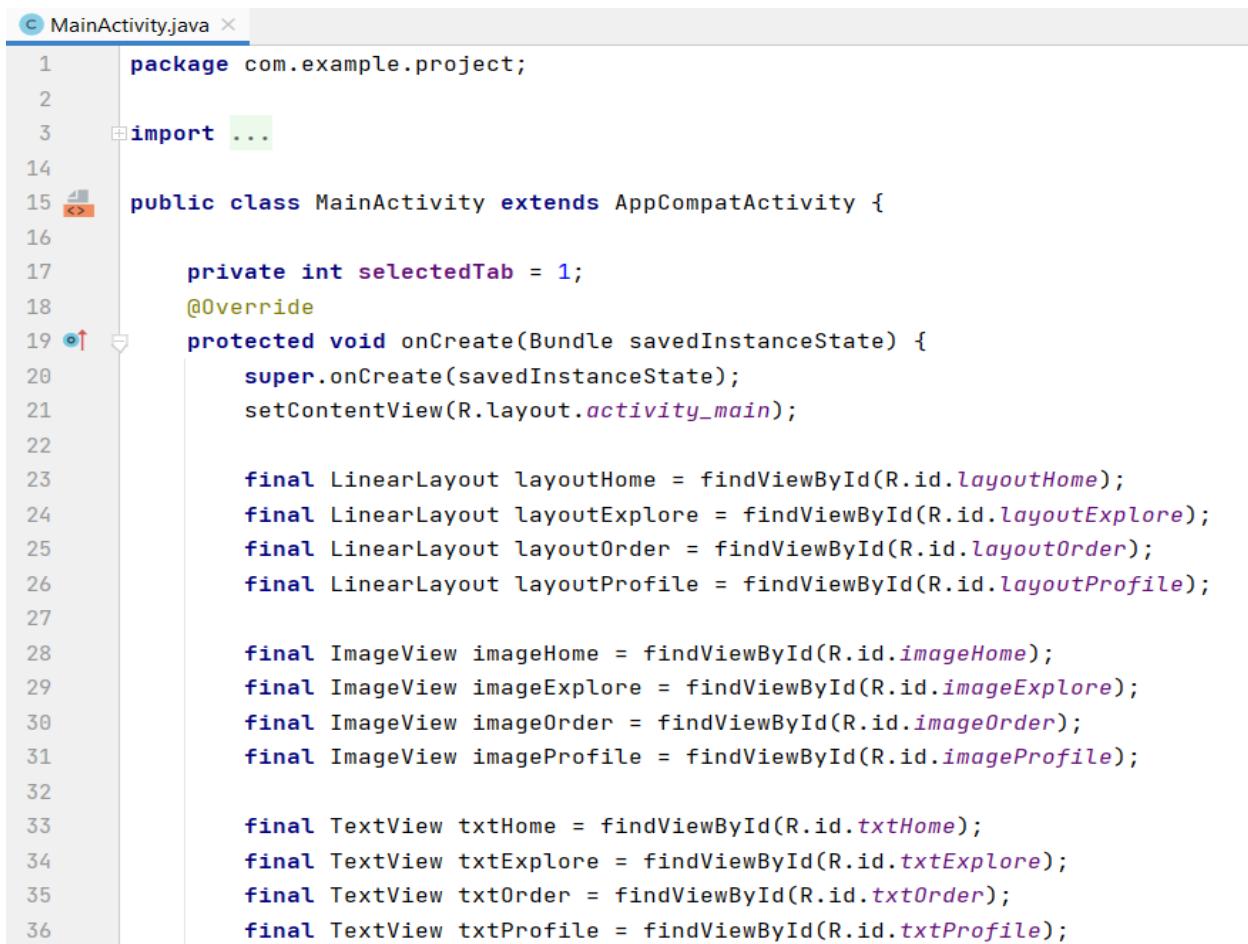
```
1 package com.example.project;
2
3 import ...
30
31 public class LoginActivity extends AppCompatActivity {
32
33     private TextInputEditText inputPass, inputUsername;
34
35     private TextView errLogin;
36
37     private String username, pass;
38
39     @Override
40     protected void onCreate(Bundle savedInstanceState) {
41         super.onCreate(savedInstanceState);
42         setContentView(R.layout.activity_login);
43         MaterialButton btnlogin = (MaterialButton) findViewById(R.id.btnLogin);
44         inputUsername = (TextInputEditText) findViewById(R.id.inputUsername);
45         inputPass = (TextInputEditText) findViewById(R.id.inputPass);
46         errLogin = (TextView) findViewById(R.id.errLogin);
47         btnlogin.setOnClickListener(view -> {
48             username = Objects.requireNonNull(inputUsername.getText()).toString();
49             pass = Objects.requireNonNull(inputPass.getText()).toString();
50
51             String url = "http://192.168.100.4/ezgo/router.php";
52             RequestQueue queue = Volley.newRequestQueue( context: this );
53             Gson gson = new Gson();
```

```
 LoginActivity.java
55     Map<String, Object> params = new HashMap<>();
56     params.put( "controller", "login");
57     params.put( "method", "login");
58     params.put( "userID", username);
59     params.put( "uPassword", pass);
60
61     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, url,
62         new JSONObject(params),
63         new Response.Listener<JSONObject>() {
64             @Override
65             public void onResponse(JSONObject response) {
66                 ResponseOneObject<User> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObject<User>>() {}.getType());
67                 boolean success = resp.isSuccess();
68                 User user = resp.getData();
69                 Log.d( tag: "Ezgo", msg: "Response: " + Boolean.toString(success));
70
71                 if (success == true) {
72                     if (errLogin != null) {
73                         errLogin.setVisibility(View.GONE);
74                     }
75                     try {
76                         internalDB db = new internalDB( context: LoginActivity.this);
77                         db.createUser(user);
78
79                         Log.d( tag: "Ezgo", msg: "SQLite created");
80
81                         Sharedpreferences preferences = getSharedpreferences( name: "ezgo", MODE_PRIVATE);
82                         Sharedpreferences.Editor editor = preferences.edit();
83                         editor.putString( "user", user.toJson());
84                         editor.apply();
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115 }
```

```
 LoginActivity.java
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115 }
```

Kode java LoginActivity di atas merupakan implementasi dari *LoginActivity* dalam aplikasi Android yang digunakan untuk mengelola proses login pengguna. Aktivitas ini menginisialisasi elemen UI seperti *TextInputEditText* untuk username dan password, serta *TextView* untuk menampilkan pesan kesalahan login. Ketika tombol login ditekan, aplikasi mengambil input username dan password, lalu mengirimkan permintaan HTTP POST menggunakan *Volley* ke server dengan URL *router.php* untuk memverifikasi kredensial pengguna. Jika login berhasil, data pengguna disimpan dalam *SharedPreferences* dan *SQLite* menggunakan kelas *internalDB*, dan pengguna diarahkan ke *MainActivity*. Jika login gagal, pesan kesalahan ditampilkan. Kode ini mengimplementasikan fitur autentikasi pengguna dengan mengirimkan data ke server dan menampilkan hasilnya melalui antarmuka pengguna, memastikan keamanan dan kemudahan akses aplikasi.

### 3.2.29. *MainActivity*



```
 MainActivity.java
1 package com.example.project;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     private int selectedTab = 1;
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13
14        final LinearLayout layoutHome = findViewById(R.id.layoutHome);
15        final LinearLayout layoutExplore = findViewById(R.id.layoutExplore);
16        final LinearLayout layoutOrder = findViewById(R.id.layoutOrder);
17        final LinearLayout layoutProfile = findViewById(R.id.layoutProfile);
18
19        final ImageView imageHome = findViewById(R.id.imageHome);
20        final ImageView imageExplore = findViewById(R.id.imageExplore);
21        final ImageView imageOrder = findViewById(R.id.imageOrder);
22        final ImageView imageProfile = findViewById(R.id.imageProfile);
23
24        final TextView txtHome = findViewById(R.id.txtHome);
25        final TextView txtExplore = findViewById(R.id.txtExplore);
26        final TextView txtOrder = findViewById(R.id.txtOrder);
27        final TextView txtProfile = findViewById(R.id.txtProfile);
28
29        ...
30
31        ...
32
33        ...
34
35        ...
36    }
}
```

```
>MainActivity.java
39  getSupportFragmentManager().beginTransaction()
40      .setReorderingAllowed(true)
41      .replace(R.id.fragmenContainer, HomeFragment.class, args: null)
42      .commit();
43
44 layoutHome.setOnClickListener(view -> {
45     if(selectedTab != 1){
46         getSupportFragmentManager().beginTransaction()
47             .setReorderingAllowed(true)
48             .replace(R.id.fragmenContainer, HomeFragment.class, args: null)
49             .commit();
50
51         txtExplore.setVisibility(View.GONE);
52         txtOrder.setVisibility(View.GONE);
53         txtProfile.setVisibility(View.GONE);
54
55         imageExplore.setImageResource(R.drawable.outline_explore_24);
56         imageOrder.setImageResource(R.drawable.round_list_alt_24_2);
57         imageProfile.setImageResource(R.drawable.outline_person_24);
58
59         layoutExplore.setBackgroundColor(getResources().getColor(android.R.color.transparent));
60         layoutOrder.setBackgroundColor(getResources().getColor(android.R.color.transparent));
61         layoutProfile.setBackgroundColor(getResources().getColor(android.R.color.transparent));
62
63         //Select home
64         txtHome.setVisibility(View.VISIBLE);
65         imageHome.setImageResource(R.drawable.round_home_24);
66         layoutHome.setBackground(R.drawable.round_back);
67
68         //Animation
69         ScaleAnimation scaleAnimation = new ScaleAnimation( fromX: 0.8f, toX: 1.0f, fromY: 1f, toY: 1f, Animation.RELATIVE_TO_SELF, pivotXValue: 0.0f, Animation.RELATIVE_TO_SELF, pivotYValue: 0.0f );
70         scaleAnimation.setFillAfter(true);
71         layoutHome.startAnimation(scaleAnimation);
72
73         selectedTab = 1;
74     }
75 });
76
77 layoutExplore.setOnClickListener(view -> {
78     if(selectedTab != 2){
79         getSupportFragmentManager().beginTransaction()
80             .setReorderingAllowed(true)
81             .replace(R.id.fragmenContainer, ExploreFragment.class, args: null)
82             .commit();
83
84         txtHome.setVisibility(View.GONE);
85         txtOrder.setVisibility(View.GONE);
86         txtProfile.setVisibility(View.GONE);
87
88         imageHome.setImageResource(R.drawable.outline_home_24);
89         imageOrder.setImageResource(R.drawable.round_list_alt_24_2);
90         imageProfile.setImageResource(R.drawable.outline_person_24);
91
92         layoutHome.setBackgroundColor(getResources().getColor(android.R.color.transparent));
93         layoutOrder.setBackgroundColor(getResources().getColor(android.R.color.transparent));
94         layoutProfile.setBackgroundColor(getResources().getColor(android.R.color.transparent));
95
96         //Select home
97         txtExplore.setVisibility(View.VISIBLE);
98         imageExplore.setImageResource(R.drawable.round_explore_24);
99         layoutExplore.setBackground(R.drawable.round_back);
```

```
>MainActivity.java X 14 ^ v
101 //Animation
102 ScaleAnimation scaleAnimation = new ScaleAnimation( fromX: 0.8f, toX: 1.0f, fromY: 1f, toY: 1f, Animation.RELATIVE_TO_SELF, pivotXValue: 1.0f, Animation.RELATIVE_TO_SELF, pivotYValue: 0.0f );
103 scaleAnimation.setDuration(200);
104 scaleAnimation.setFillAfter(true);
105 layoutExplore.startAnimation(scaleAnimation);
106
107 selectedTab = 2;
108 }
109 });
110
111 layoutOrder.setOnClickListener(view -> {
112 if(selectedTab != 3){
113     getSupportFragmentManager().beginTransaction()
114         .setReorderingAllowed(true)
115         .replace(R.id.fragmentContainer, OrderFragment.class, args: null)
116         .commit();
117
118     txtExplore.setVisibility(View.GONE);
119     txtHome.setVisibility(View.GONE);
120     txtProfile.setVisibility(View.GONE);
121
122     imageExplore.setImageResource(R.drawable.outline_explore_24);
123     imageHome.setImageResource(R.drawable.outline_home_24);
124     imageProfile.setImageResource(R.drawable.outline_person_24);
125
126     layoutExplore.setBackgroundColor(getResources().getColor(android.R.color.transparent));
127     layoutHome.setBackgroundColor(getResources().getColor(android.R.color.transparent));
128     layoutProfile.setBackgroundColor(getResources().getColor(android.R.color.transparent));
129
130     //Select home
131     txtOrder.setVisibility(View.VISIBLE);
132
133     imageOrder.setImageResource(R.drawable.round_list_alt_24);
134     layoutOrder.setBackgroundResource(R.drawable.round_back);
135
136     //Animation
137     ScaleAnimation scaleAnimation = new ScaleAnimation( fromX: 0.8f, toX: 1.0f, fromY: 1f, toY: 1f, Animation.RELATIVE_TO_SELF, pivotXValue: 1.0f, Animation.RELATIVE_TO_SELF, pivotYValue: 0.0f );
138     scaleAnimation.setDuration(200);
139     scaleAnimation.setFillAfter(true);
140     layoutOrder.startAnimation(scaleAnimation);
141
142     selectedTab = 3;
143 }
144 });
145
146 layoutProfile.setOnClickListener(view -> {
147 if(selectedTab != 4){
148     getSupportFragmentManager().beginTransaction()
149         .setReorderingAllowed(true)
150         .replace(R.id.fragmentContainer, ProfileFragment.class, args: null)
151         .commit();
152
153     txtExplore.setVisibility(View.GONE);
154     txtOrder.setVisibility(View.GONE);
155     txtHome.setVisibility(View.GONE);
156
157     imageExplore.setImageResource(R.drawable.outline_explore_24);
158     imageOrder.setImageResource(R.drawable.round_list_alt_24_2);
159     imageHome.setImageResource(R.drawable.outline_home_24);
160
161     layoutExplore.setBackgroundColor(getResources().getColor(android.R.color.transparent));
162     layoutOrder.setBackgroundColor(getResources().getColor(android.R.color.transparent));
163     layoutHome.setBackgroundColor(getResources().getColor(android.R.color.transparent));
164 }
```

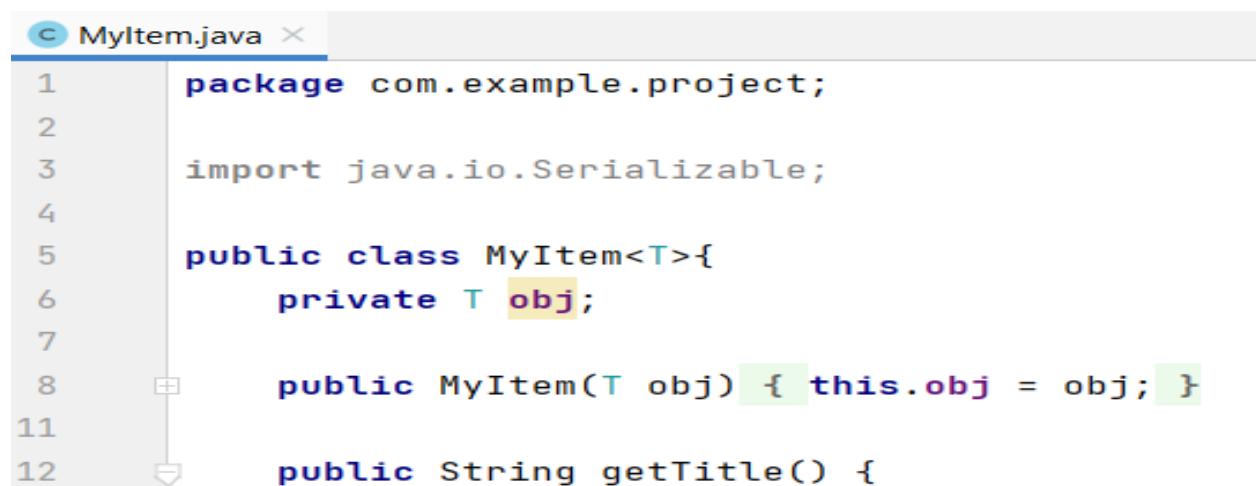
```
>MainActivity.java X 14 ^ v
132     imageOrder.setImageResource(R.drawable.round_list_alt_24);
133     layoutOrder.setBackgroundResource(R.drawable.round_back);
134
135     //Animation
136     ScaleAnimation scaleAnimation = new ScaleAnimation( fromX: 0.8f, toX: 1.0f, fromY: 1f, toY: 1f, Animation.RELATIVE_TO_SELF, pivotXValue: 1.0f, Animation.RELATIVE_TO_SELF, pivotYValue: 0.0f );
137     scaleAnimation.setDuration(200);
138     scaleAnimation.setFillAfter(true);
139     layoutOrder.startAnimation(scaleAnimation);
140
141     selectedTab = 3;
142 }
143 });
144
145 layoutProfile.setOnClickListener(view -> {
146 if(selectedTab != 4){
147     getSupportFragmentManager().beginTransaction()
148         .setReorderingAllowed(true)
149         .replace(R.id.fragmentContainer, ProfileFragment.class, args: null)
150         .commit();
151
152     txtExplore.setVisibility(View.GONE);
153     txtOrder.setVisibility(View.GONE);
154     txtHome.setVisibility(View.GONE);
155
156     imageExplore.setImageResource(R.drawable.outline_explore_24);
157     imageOrder.setImageResource(R.drawable.round_list_alt_24_2);
158     imageHome.setImageResource(R.drawable.outline_home_24);
159
160     layoutExplore.setBackgroundColor(getResources().getColor(android.R.color.transparent));
161     layoutOrder.setBackgroundColor(getResources().getColor(android.R.color.transparent));
162     layoutHome.setBackgroundColor(getResources().getColor(android.R.color.transparent));
163 }
```



```
164     //Select home
165     txtProfile.setVisibility(View.VISIBLE);
166     imageProfile.setImageResource(R.drawable.round_person_24);
167     layoutProfile.setBackgroundResource(R.drawable.round_back);
168
169     //Animation
170     ScaleAnimation scaleAnimation = new ScaleAnimation( fromX: 0.8f, toX: 1.0f, fromY: 1f, toY: 1f, Animation.RELATIVE_TO_SELF, pivotXValue: 1.0f, Animation.RELATIVE_TO_SELF, pivotYValue: 0.0f );
171     scaleAnimation.setDuration(200);
172     scaleAnimation.setFillAfter(true);
173     layoutProfile.startAnimation(scaleAnimation);
174
175     selectedTab = 4;
176
177 );
```

Kode java *MainActivity* di atas merupakan implementasi dari *MainActivity* dalam aplikasi Android yang berfungsi sebagai aktivitas utama yang mengelola navigasi antar-fragment menggunakan tab di bagian bawah layar. Saat aktivitas ini dibuat, elemen UI seperti *LinearLayout*, *ImageView*, dan *TextView* untuk setiap tab diinisialisasi. Tab-tab tersebut adalah *Home*, *Explore*, *Order*, dan *Profile*. Metode *setOnItemClickListener* diterapkan pada setiap tab untuk mengganti fragment yang ditampilkan di dalam *fragmenContainer* dengan fragment yang sesuai ketika tab ditekan. Selain mengganti fragment, setiap klik juga mengubah tampilan visual tab yang aktif, termasuk menampilkan teks dan mengganti ikon serta menerapkan animasi skala. Kode ini mengimplementasikan fitur navigasi antar-fragment yang mudah diakses dan memberikan umpan balik visual yang dinamis.

### **3.2.30. *MyItem***



```
1 package com.example.project;
2
3 import java.io.Serializable;
4
5 public class MyItem<T>{
6     private T obj;
7
8     public MyItem(T obj) { this.obj = obj; }
11
12     public String getTitle() {
```

```

  MyItem.java ×
13     if (obj instanceof location) {
14         return ((location) obj).lName;
15     }else if(obj instanceof ticket){
16         return ((ticket) obj).tcName;
17     }else if(obj instanceof hotel){
18         return ((hotel) obj).hName;
19     }else if(obj instanceof tour){
20         return ((tour) obj).tpName;
21     }else{
22         return null;
23     }
24 }
25
26     public String getImage() {
27         if (obj instanceof location) {
28             return ((location) obj).lImage;
29         }else if(obj instanceof ticket){
30             return ((ticket) obj).tcImage;
31         }else if(obj instanceof hotel){
32             return ((hotel) obj).hImage;
33         }else if(obj instanceof tour){
34             return ((tour) obj).tpImage;
35         }else{
36             return null;
37         }
38     }
39
40     public Class<?> getObjectType() { return obj.getClass(); }
41
42     public T getObj() { return obj; }
43
44 }
45
46
47 }

```

Kode java MyItem di atas merupakan implementasi dari kelas *MyItem* dalam aplikasi Android yang digunakan untuk mengelola objek generik dengan tipe yang berbeda seperti *location*, *ticket*, *hotel*, dan *tour*. Kelas ini menyimpan objek *obj* dan menyediakan metode untuk mendapatkan judul (*getTitle*), gambar (*getImage*), tipe objek (*getObjectType*), dan objek itu sendiri (*getObj*). Metode *getTitle* dan *getImage* menggunakan instance checking untuk mengembalikan nilai yang sesuai berdasarkan tipe objek yang disimpan. Kode ini mengimplementasikan fitur yang memungkinkan penanganan berbagai tipe objek dalam satu kelas, sehingga memudahkan pengelolaan dan penggunaan data yang berbeda dalam aplikasi, terutama dalam konteks tampilan yang seragam seperti dalam adapter untuk *RecyclerView*.

### 3.2.31. OrderFragment

```
OrderFragment.java
1 package com.example.project;
2
3 import ...
37
38 public class OrderFragment extends Fragment {
39     RecyclerView ticketRecyclerView, hotelRecyclerView, tourRecyclerView;
40     AdapterViewTicket adapterViewTicket;
41     AdapterViewHotel adapterViewHotel;
42     AdapterViewTour adapterViewTour;
43     SearchView btnSearch;
44
45     @Nullable
46     @Override
47     public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
48         View v = inflater.inflate(R.layout.activity_order, container, false);
49         Sharedpreferences preferences = getActivity().getSharedpreferences(name: "ezgo", Context.MODE_PRIVATE);
50         String userJson = preferences.getString(s: "user", s1: null);
51         User user = new Gson().fromJson(userJson, User.class);
52         Log.d(tag: "Ezgo", msg: "uName3: " + user.uName);
53
54         btnSearch = v.findViewById(R.id.SearchBar);
55
56         String urlReq = "http://192.168.100.4/ezgo/router.php";
57
58         RequestQueue queue = Volley.newRequestQueue(getActivity());
59         Gson gson = new Gson();
60
61         Map<String, Object> params = new HashMap<>();
62         params.put(k: "controller", v: "transaction");
63         params.put(k: "method", v: "transAll");
64         params.put(k: "userID", user.userID);
```

```
Orderfragment.java
66 JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
67     new JSONObject(params),
68     new Response.Listener<JSONObject>() {
69         @Override
70         public void onResponse(JSONObject response) {
71             Log.d(tag: "Ezgo", msg: "ResponseHome: " + response);
72             ResponseFourObjectList<String, ticket, hotel, tour> resp = gson.fromJson(response.toString(), new TypeToken<ResponseFourObjectList<String, ticket, hotel, tour>>() {
73                 boolean success = resp.isSuccess();
74                 List<String> ids = resp.getData1();
75                 List<ticket> tixs = resp.getData2();
76                 List<hotel> htls = resp.getData3();
77                 List<tour> trps = resp.getData4();
78
79                 if (success == true) {
80                     try {
81                         List<ticket> tickets = new ArrayList<>();
82                         List<hotel> hotels = new ArrayList<>();
83                         List<tour> tours = new ArrayList<>();
84
85                         for (String id : ids) {
86                             for (ticket tix : tixs) {
87                                 if(tix.productID.equals(id)){
88                                     tickets.add(tix);
89                                 }
90                             }
91                             for (hotel htl : htls) {
92                                 if(htl.productID.equals(id)){
93                                     hotels.add(htl);
94                                 }
95                             }
96                             for (tour trp : trps) {
```

```

  OrderFragment.java x
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
  OrderFragment.java x
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144

```

```

    if(trp.productID.equals(id)){
        tours.add(trp);
    }
}

// Set up RecyclerView and Adapter for tickets
ticketRecyclerView = v.findViewById(R.id.hist_orderTicket);
adapterViewTicket = new AdapterViewTicket(getActivity(), tickets);
ticketRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
ticketRecyclerView.setAdapter(adapterViewTicket);

// Set up RecyclerView and Adapter for hotels
hotelRecyclerView = v.findViewById(R.id.hist_orderHotel);
adapterViewHotel = new AdapterViewHotel(getActivity(), hotels);
hotelRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
hotelRecyclerView.setAdapter(adapterViewHotel);

// Set up RecyclerView and Adapter for tours
tourRecyclerView = v.findViewById(R.id.hist_orderTour);
adapterViewTour = new AdapterViewTour(getActivity(), tours);
tourRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
tourRecyclerView.setAdapter(adapterViewTour);

} catch (Exception e) {
    Log.e( tag: "Ezgo", msg: "Error: " + e);
}
}

},
new Response.ErrorListener() {

    @Override
    public void onErrorResponse(VolleyError error) {
        Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
        String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
        Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
    }
);
queue.add(jsonObjectRequest);

btnSearch.setOnClickListener(view -> {
    Intent i = new Intent(getActivity(),SearchActivity.class);
    startActivity(i);
});

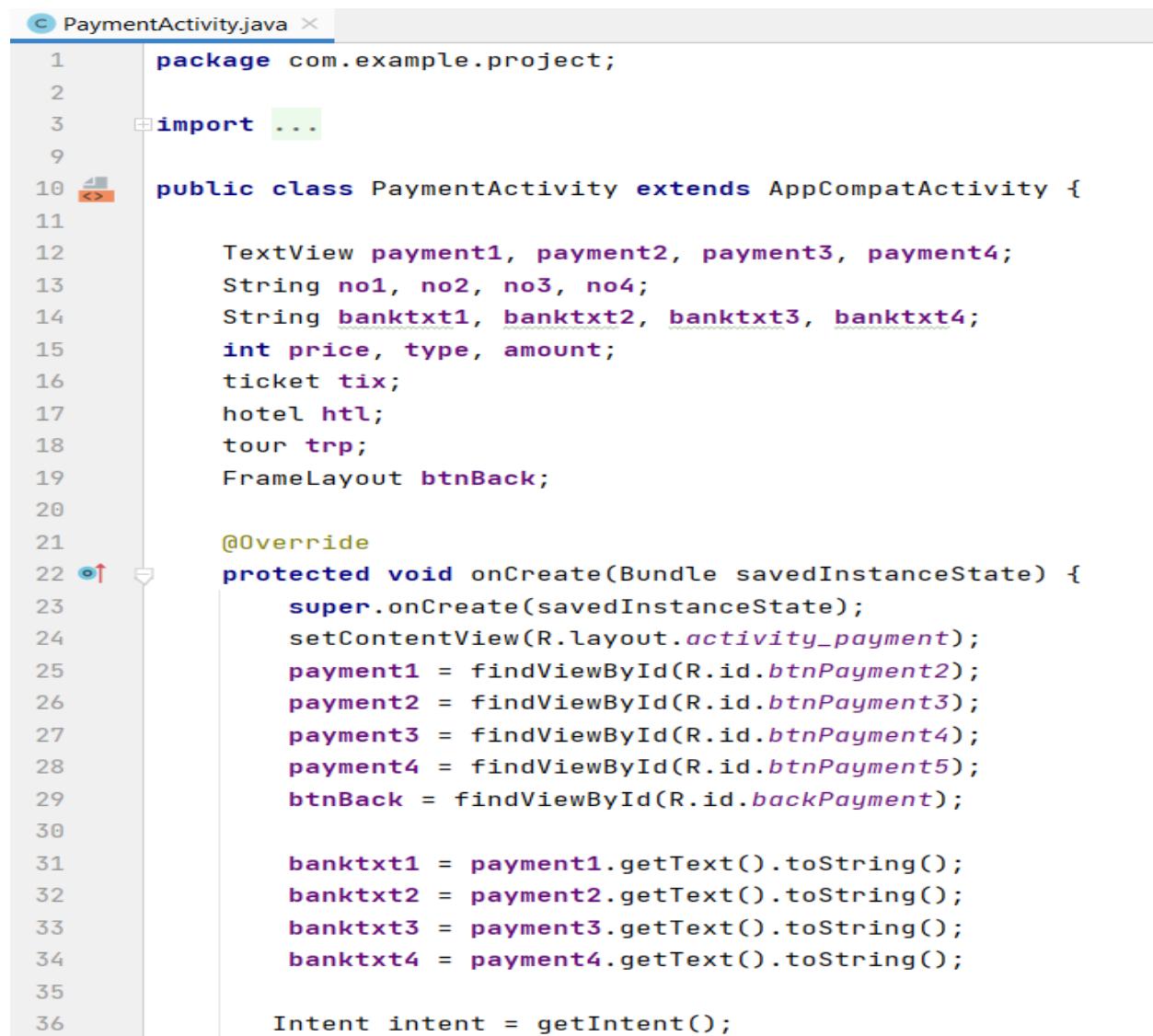
return v;
}
}

```

Kode java OrderFragment di atas merupakan implementasi dari *OrderFragment* dalam aplikasi Android yang digunakan untuk menampilkan riwayat pesanan pengguna yang mencakup tiket, hotel, dan tur. Ketika fragment ini dibuat, elemen UI seperti *RecyclerView* dan *SearchView* diinisialisasi. Data pengguna diambil dari *SharedPreferences*, dan permintaan HTTP POST

dikirim ke server menggunakan *Volley* untuk mengambil data pesanan pengguna berdasarkan *userID*. Data yang diterima dikonversi dari JSON menjadi objek *String*, *ticket*, *hotel*, dan *tour* menggunakan *Gson*. Setelah data diterima, fragment ini mengatur *RecyclerView* dan *adapter* untuk menampilkan daftar tiket, hotel, dan tur yang telah dipesan oleh pengguna. Tombol pencarian (*btnSearch*) memungkinkan pengguna untuk melakukan pencarian lebih lanjut dengan membuka *SearchActivity*. Kode ini mengimplementasikan fitur untuk menampilkan dan mengelola riwayat pesanan pengguna dengan antarmuka yang terstruktur dan mudah diakses.

### ***3.2.32. PaymentActivity***



```
1 package com.example.project;
2
3 import ...
4
5
6 public class PaymentActivity extends AppCompatActivity {
7
8     TextView payment1, payment2, payment3, payment4;
9     String no1, no2, no3, no4;
10    String banktxt1, banktxt2, banktxt3, banktxt4;
11    int price, type, amount;
12    ticket tix;
13    hotel htl;
14    tour trp;
15    FrameLayout btnBack;
16
17    @Override
18    protected void onCreate(Bundle savedInstanceState) {
19        super.onCreate(savedInstanceState);
20        setContentView(R.layout.activity_payment);
21        payment1 = findViewById(R.id.btnPayment2);
22        payment2 = findViewById(R.id.btnPayment3);
23        payment3 = findViewById(R.id.btnPayment4);
24        payment4 = findViewById(R.id.btnPayment5);
25        btnBack = findViewById(R.id.backPayment);
26
27        banktxt1 = payment1.getText().toString();
28        banktxt2 = payment2.getText().toString();
29        banktxt3 = payment3.getText().toString();
30        banktxt4 = payment4.getText().toString();
31
32        Intent intent = getIntent();
33    }
34
35
36 }
```

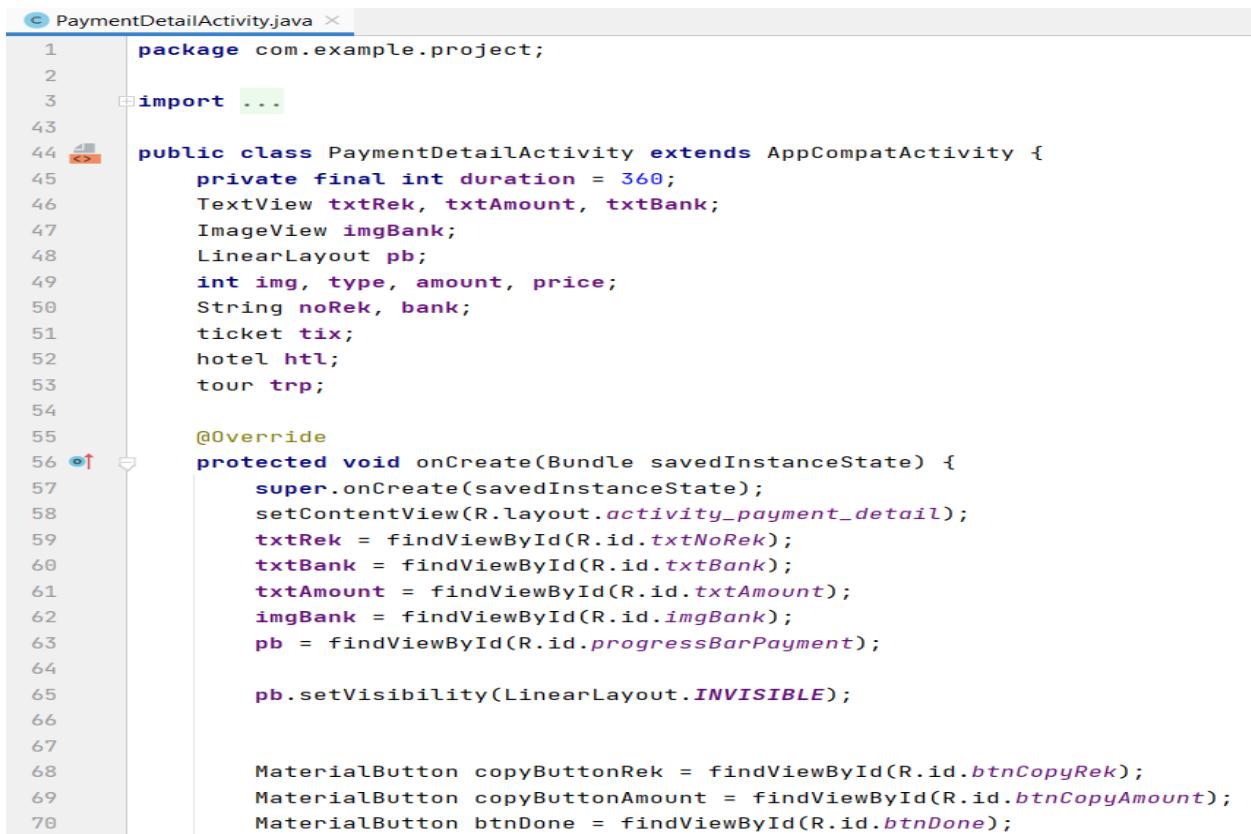
```
PaymentActivity.java
37     price = intent.getIntExtra( name: "price", defaultValue: 0);
38     amount = intent.getIntExtra( name: "amount", defaultValue: 0);
39
40     if(intent.getSerializableExtra( name: "object") instanceof ticket){
41         tix = (ticket) intent.getSerializableExtra( name: "object");
42         type = 1;
43     }else if(intent.getSerializableExtra( name: "object") instanceof hotel){
44         htl = (hotel) intent.getSerializableExtra( name: "object");
45         type = 2;
46     }else if(intent.getSerializableExtra( name: "object") instanceof tour){
47         trp = (tour) intent.getSerializableExtra( name: "object");
48         type = 3;
49     }
50
51     no1 = "A";
52     no2 = "B";
53     no3 = "C";
54     no4 = "D";
55
56     btnBack.setOnClickListener(view -> onBackPressed());
57     payment1.setOnClickListener(view -> {
58         Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
59         Bundle bundle1 = new Bundle();
60         bundle1.putString("norek", no1);
61         bundle1.putString("bank", banktxt1);
62         bundle1.putInt("price", price);
63         bundle1.putInt("amount", amount);
64         bundle1.putInt("draw", R.drawable.bca);
65
66         switch (type){
67             case 1:
68                 bundle1.putSerializable("object", tix);
69                 break;
70             case 2:
71                 bundle1.putSerializable("object", htl);
72                 break;
73             case 3:
74                 bundle1.putSerializable("object", trp);
75                 break;
76         }
77
78         i.putExtras(bundle1);
79         startActivity(i);
80     });
81     payment2.setOnClickListener(view -> {
82         Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
83         Bundle bundle2 = new Bundle();
84         bundle2.putString("norek", no2);
85         bundle2.putString("bank", banktxt2);
86         bundle2.putInt("price", price);
87         bundle2.putInt("amount", amount);
88
89         switch (type){
90             case 1:
91                 bundle2.putSerializable("object", tix);
92                 break;
93             case 2:
94                 bundle2.putSerializable("object", htl);
95                 break;
96             case 3:
97                 bundle2.putSerializable("object", trp);
98                 break;
99         }
100    });
101
102    payment3.setOnClickListener(view -> {
103        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
104        Bundle bundle3 = new Bundle();
105        bundle3.putString("norek", no3);
106        bundle3.putString("bank", banktxt3);
107        bundle3.putInt("price", price);
108        bundle3.putInt("amount", amount);
109
110        switch (type){
111            case 1:
112                bundle3.putSerializable("object", tix);
113                break;
114            case 2:
115                bundle3.putSerializable("object", htl);
116                break;
117            case 3:
118                bundle3.putSerializable("object", trp);
119                break;
120        }
121
122        i.putExtras(bundle3);
123        startActivity(i);
124    });
125
126    payment4.setOnClickListener(view -> {
127        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
128        Bundle bundle4 = new Bundle();
129        bundle4.putString("norek", no4);
130        bundle4.putString("bank", banktxt4);
131        bundle4.putInt("price", price);
132        bundle4.putInt("amount", amount);
133
134        switch (type){
135            case 1:
136                bundle4.putSerializable("object", tix);
137                break;
138            case 2:
139                bundle4.putSerializable("object", htl);
140                break;
141            case 3:
142                bundle4.putSerializable("object", trp);
143                break;
144        }
145
146        i.putExtras(bundle4);
147        startActivity(i);
148    });
149
150    payment5.setOnClickListener(view -> {
151        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
152        Bundle bundle5 = new Bundle();
153        bundle5.putString("norek", no5);
154        bundle5.putString("bank", banktxt5);
155        bundle5.putInt("price", price);
156        bundle5.putInt("amount", amount);
157
158        switch (type){
159            case 1:
160                bundle5.putSerializable("object", tix);
161                break;
162            case 2:
163                bundle5.putSerializable("object", htl);
164                break;
165            case 3:
166                bundle5.putSerializable("object", trp);
167                break;
168        }
169
170        i.putExtras(bundle5);
171        startActivity(i);
172    });
173
174    payment6.setOnClickListener(view -> {
175        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
176        Bundle bundle6 = new Bundle();
177        bundle6.putString("norek", no6);
178        bundle6.putString("bank", banktxt6);
179        bundle6.putInt("price", price);
180        bundle6.putInt("amount", amount);
181
182        switch (type){
183            case 1:
184                bundle6.putSerializable("object", tix);
185                break;
186            case 2:
187                bundle6.putSerializable("object", htl);
188                break;
189            case 3:
190                bundle6.putSerializable("object", trp);
191                break;
192        }
193
194        i.putExtras(bundle6);
195        startActivity(i);
196    });
197
198    payment7.setOnClickListener(view -> {
199        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
200        Bundle bundle7 = new Bundle();
201        bundle7.putString("norek", no7);
202        bundle7.putString("bank", banktxt7);
203        bundle7.putInt("price", price);
204        bundle7.putInt("amount", amount);
205
206        switch (type){
207            case 1:
208                bundle7.putSerializable("object", tix);
209                break;
210            case 2:
211                bundle7.putSerializable("object", htl);
212                break;
213            case 3:
214                bundle7.putSerializable("object", trp);
215                break;
216        }
217
218        i.putExtras(bundle7);
219        startActivity(i);
220    });
221
222    payment8.setOnClickListener(view -> {
223        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
224        Bundle bundle8 = new Bundle();
225        bundle8.putString("norek", no8);
226        bundle8.putString("bank", banktxt8);
227        bundle8.putInt("price", price);
228        bundle8.putInt("amount", amount);
229
230        switch (type){
231            case 1:
232                bundle8.putSerializable("object", tix);
233                break;
234            case 2:
235                bundle8.putSerializable("object", htl);
236                break;
237            case 3:
238                bundle8.putSerializable("object", trp);
239                break;
240        }
241
242        i.putExtras(bundle8);
243        startActivity(i);
244    });
245
246    payment9.setOnClickListener(view -> {
247        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
248        Bundle bundle9 = new Bundle();
249        bundle9.putString("norek", no9);
250        bundle9.putString("bank", banktxt9);
251        bundle9.putInt("price", price);
252        bundle9.putInt("amount", amount);
253
254        switch (type){
255            case 1:
256                bundle9.putSerializable("object", tix);
257                break;
258            case 2:
259                bundle9.putSerializable("object", htl);
260                break;
261            case 3:
262                bundle9.putSerializable("object", trp);
263                break;
264        }
265
266        i.putExtras(bundle9);
267        startActivity(i);
268    });
269
270    payment10.setOnClickListener(view -> {
271        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
272        Bundle bundle10 = new Bundle();
273        bundle10.putString("norek", no10);
274        bundle10.putString("bank", banktxt10);
275        bundle10.putInt("price", price);
276        bundle10.putInt("amount", amount);
277
278        switch (type){
279            case 1:
280                bundle10.putSerializable("object", tix);
281                break;
282            case 2:
283                bundle10.putSerializable("object", htl);
284                break;
285            case 3:
286                bundle10.putSerializable("object", trp);
287                break;
288        }
289
290        i.putExtras(bundle10);
291        startActivity(i);
292    });
293
294    payment11.setOnClickListener(view -> {
295        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
296        Bundle bundle11 = new Bundle();
297        bundle11.putString("norek", no11);
298        bundle11.putString("bank", banktxt11);
299        bundle11.putInt("price", price);
300        bundle11.putInt("amount", amount);
301
302        switch (type){
303            case 1:
304                bundle11.putSerializable("object", tix);
305                break;
306            case 2:
307                bundle11.putSerializable("object", htl);
308                break;
309            case 3:
310                bundle11.putSerializable("object", trp);
311                break;
312        }
313
314        i.putExtras(bundle11);
315        startActivity(i);
316    });
317
318    payment12.setOnClickListener(view -> {
319        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
320        Bundle bundle12 = new Bundle();
321        bundle12.putString("norek", no12);
322        bundle12.putString("bank", banktxt12);
323        bundle12.putInt("price", price);
324        bundle12.putInt("amount", amount);
325
326        switch (type){
327            case 1:
328                bundle12.putSerializable("object", tix);
329                break;
330            case 2:
331                bundle12.putSerializable("object", htl);
332                break;
333            case 3:
334                bundle12.putSerializable("object", trp);
335                break;
336        }
337
338        i.putExtras(bundle12);
339        startActivity(i);
340    });
341
342    payment13.setOnClickListener(view -> {
343        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
344        Bundle bundle13 = new Bundle();
345        bundle13.putString("norek", no13);
346        bundle13.putString("bank", banktxt13);
347        bundle13.putInt("price", price);
348        bundle13.putInt("amount", amount);
349
350        switch (type){
351            case 1:
352                bundle13.putSerializable("object", tix);
353                break;
354            case 2:
355                bundle13.putSerializable("object", htl);
356                break;
357            case 3:
358                bundle13.putSerializable("object", trp);
359                break;
360        }
361
362        i.putExtras(bundle13);
363        startActivity(i);
364    });
365
366    payment14.setOnClickListener(view -> {
367        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
368        Bundle bundle14 = new Bundle();
369        bundle14.putString("norek", no14);
370        bundle14.putString("bank", banktxt14);
371        bundle14.putInt("price", price);
372        bundle14.putInt("amount", amount);
373
374        switch (type){
375            case 1:
376                bundle14.putSerializable("object", tix);
377                break;
378            case 2:
379                bundle14.putSerializable("object", htl);
380                break;
381            case 3:
382                bundle14.putSerializable("object", trp);
383                break;
384        }
385
386        i.putExtras(bundle14);
387        startActivity(i);
388    });
389
390    payment15.setOnClickListener(view -> {
391        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
392        Bundle bundle15 = new Bundle();
393        bundle15.putString("norek", no15);
394        bundle15.putString("bank", banktxt15);
395        bundle15.putInt("price", price);
396        bundle15.putInt("amount", amount);
397
398        switch (type){
399            case 1:
400                bundle15.putSerializable("object", tix);
401                break;
402            case 2:
403                bundle15.putSerializable("object", htl);
404                break;
405            case 3:
406                bundle15.putSerializable("object", trp);
407                break;
408        }
409
410        i.putExtras(bundle15);
411        startActivity(i);
412    });
413
414    payment16.setOnClickListener(view -> {
415        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
416        Bundle bundle16 = new Bundle();
417        bundle16.putString("norek", no16);
418        bundle16.putString("bank", banktxt16);
419        bundle16.putInt("price", price);
420        bundle16.putInt("amount", amount);
421
422        switch (type){
423            case 1:
424                bundle16.putSerializable("object", tix);
425                break;
426            case 2:
427                bundle16.putSerializable("object", htl);
428                break;
429            case 3:
430                bundle16.putSerializable("object", trp);
431                break;
432        }
433
434        i.putExtras(bundle16);
435        startActivity(i);
436    });
437
438    payment17.setOnClickListener(view -> {
439        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
440        Bundle bundle17 = new Bundle();
441        bundle17.putString("norek", no17);
442        bundle17.putString("bank", banktxt17);
443        bundle17.putInt("price", price);
444        bundle17.putInt("amount", amount);
445
446        switch (type){
447            case 1:
448                bundle17.putSerializable("object", tix);
449                break;
450            case 2:
451                bundle17.putSerializable("object", htl);
452                break;
453            case 3:
454                bundle17.putSerializable("object", trp);
455                break;
456        }
457
458        i.putExtras(bundle17);
459        startActivity(i);
460    });
461
462    payment18.setOnClickListener(view -> {
463        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
464        Bundle bundle18 = new Bundle();
465        bundle18.putString("norek", no18);
466        bundle18.putString("bank", banktxt18);
467        bundle18.putInt("price", price);
468        bundle18.putInt("amount", amount);
469
470        switch (type){
471            case 1:
472                bundle18.putSerializable("object", tix);
473                break;
474            case 2:
475                bundle18.putSerializable("object", htl);
476                break;
477            case 3:
478                bundle18.putSerializable("object", trp);
479                break;
480        }
481
482        i.putExtras(bundle18);
483        startActivity(i);
484    });
485
486    payment19.setOnClickListener(view -> {
487        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
488        Bundle bundle19 = new Bundle();
489        bundle19.putString("norek", no19);
490        bundle19.putString("bank", banktxt19);
491        bundle19.putInt("price", price);
492        bundle19.putInt("amount", amount);
493
494        switch (type){
495            case 1:
496                bundle19.putSerializable("object", tix);
497                break;
498            case 2:
499                bundle19.putSerializable("object", htl);
500                break;
501            case 3:
502                bundle19.putSerializable("object", trp);
503                break;
504        }
505
506        i.putExtras(bundle19);
507        startActivity(i);
508    });
509
510    payment20.setOnClickListener(view -> {
511        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
512        Bundle bundle20 = new Bundle();
513        bundle20.putString("norek", no20);
514        bundle20.putString("bank", banktxt20);
515        bundle20.putInt("price", price);
516        bundle20.putInt("amount", amount);
517
518        switch (type){
519            case 1:
520                bundle20.putSerializable("object", tix);
521                break;
522            case 2:
523                bundle20.putSerializable("object", htl);
524                break;
525            case 3:
526                bundle20.putSerializable("object", trp);
527                break;
528        }
529
530        i.putExtras(bundle20);
531        startActivity(i);
532    });
533
534    payment21.setOnClickListener(view -> {
535        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
536        Bundle bundle21 = new Bundle();
537        bundle21.putString("norek", no21);
538        bundle21.putString("bank", banktxt21);
539        bundle21.putInt("price", price);
540        bundle21.putInt("amount", amount);
541
542        switch (type){
543            case 1:
544                bundle21.putSerializable("object", tix);
545                break;
546            case 2:
547                bundle21.putSerializable("object", htl);
548                break;
549            case 3:
550                bundle21.putSerializable("object", trp);
551                break;
552        }
553
554        i.putExtras(bundle21);
555        startActivity(i);
556    });
557
558    payment22.setOnClickListener(view -> {
559        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
560        Bundle bundle22 = new Bundle();
561        bundle22.putString("norek", no22);
562        bundle22.putString("bank", banktxt22);
563        bundle22.putInt("price", price);
564        bundle22.putInt("amount", amount);
565
566        switch (type){
567            case 1:
568                bundle22.putSerializable("object", tix);
569                break;
570            case 2:
571                bundle22.putSerializable("object", htl);
572                break;
573            case 3:
574                bundle22.putSerializable("object", trp);
575                break;
576        }
577
578        i.putExtras(bundle22);
579        startActivity(i);
580    });
581
582    payment23.setOnClickListener(view -> {
583        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
584        Bundle bundle23 = new Bundle();
585        bundle23.putString("norek", no23);
586        bundle23.putString("bank", banktxt23);
587        bundle23.putInt("price", price);
588        bundle23.putInt("amount", amount);
589
590        switch (type){
591            case 1:
592                bundle23.putSerializable("object", tix);
593                break;
594            case 2:
595                bundle23.putSerializable("object", htl);
596                break;
597            case 3:
598                bundle23.putSerializable("object", trp);
599                break;
600        }
601
602        i.putExtras(bundle23);
603        startActivity(i);
604    });
605
606    payment24.setOnClickListener(view -> {
607        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
608        Bundle bundle24 = new Bundle();
609        bundle24.putString("norek", no24);
610        bundle24.putString("bank", banktxt24);
611        bundle24.putInt("price", price);
612        bundle24.putInt("amount", amount);
613
614        switch (type){
615            case 1:
616                bundle24.putSerializable("object", tix);
617                break;
618            case 2:
619                bundle24.putSerializable("object", htl);
620                break;
621            case 3:
622                bundle24.putSerializable("object", trp);
623                break;
624        }
625
626        i.putExtras(bundle24);
627        startActivity(i);
628    });
629
630    payment25.setOnClickListener(view -> {
631        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
632        Bundle bundle25 = new Bundle();
633        bundle25.putString("norek", no25);
634        bundle25.putString("bank", banktxt25);
635        bundle25.putInt("price", price);
636        bundle25.putInt("amount", amount);
637
638        switch (type){
639            case 1:
640                bundle25.putSerializable("object", tix);
641                break;
642            case 2:
643                bundle25.putSerializable("object", htl);
644                break;
645            case 3:
646                bundle25.putSerializable("object", trp);
647                break;
648        }
649
650        i.putExtras(bundle25);
651        startActivity(i);
652    });
653
654    payment26.setOnClickListener(view -> {
655        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
656        Bundle bundle26 = new Bundle();
657        bundle26.putString("norek", no26);
658        bundle26.putString("bank", banktxt26);
659        bundle26.putInt("price", price);
660        bundle26.putInt("amount", amount);
661
662        switch (type){
663            case 1:
664                bundle26.putSerializable("object", tix);
665                break;
666            case 2:
667                bundle26.putSerializable("object", htl);
668                break;
669            case 3:
670                bundle26.putSerializable("object", trp);
671                break;
672        }
673
674        i.putExtras(bundle26);
675        startActivity(i);
676    });
677
678    payment27.setOnClickListener(view -> {
679        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
680        Bundle bundle27 = new Bundle();
681        bundle27.putString("norek", no27);
682        bundle27.putString("bank", banktxt27);
683        bundle27.putInt("price", price);
684        bundle27.putInt("amount", amount);
685
686        switch (type){
687            case 1:
688                bundle27.putSerializable("object", tix);
689                break;
690            case 2:
691                bundle27.putSerializable("object", htl);
692                break;
693            case 3:
694                bundle27.putSerializable("object", trp);
695                break;
696        }
697
698        i.putExtras(bundle27);
699        startActivity(i);
700    });
701
702    payment28.setOnClickListener(view -> {
703        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
704        Bundle bundle28 = new Bundle();
705        bundle28.putString("norek", no28);
706        bundle28.putString("bank", banktxt28);
707        bundle28.putInt("price", price);
708        bundle28.putInt("amount", amount);
709
710        switch (type){
711            case 1:
712                bundle28.putSerializable("object", tix);
713                break;
714            case 2:
715                bundle28.putSerializable("object", htl);
716                break;
717            case 3:
718                bundle28.putSerializable("object", trp);
719                break;
720        }
721
722        i.putExtras(bundle28);
723        startActivity(i);
724    });
725
726    payment29.setOnClickListener(view -> {
727        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
728        Bundle bundle29 = new Bundle();
729        bundle29.putString("norek", no29);
730        bundle29.putString("bank", banktxt29);
731        bundle29.putInt("price", price);
732        bundle29.putInt("amount", amount);
733
734        switch (type){
735            case 1:
736                bundle29.putSerializable("object", tix);
737                break;
738            case 2:
739                bundle29.putSerializable("object", htl);
740                break;
741            case 3:
742                bundle29.putSerializable("object", trp);
743                break;
744        }
745
746        i.putExtras(bundle29);
747        startActivity(i);
748    });
749
750    payment30.setOnClickListener(view -> {
751        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
752        Bundle bundle30 = new Bundle();
753        bundle30.putString("norek", no30);
754        bundle30.putString("bank", banktxt30);
755        bundle30.putInt("price", price);
756        bundle30.putInt("amount", amount);
757
758        switch (type){
759            case 1:
760                bundle30.putSerializable("object", tix);
761                break;
762            case 2:
763                bundle30.putSerializable("object", htl);
764                break;
765            case 3:
766                bundle30.putSerializable("object", trp);
767                break;
768        }
769
770        i.putExtras(bundle30);
771        startActivity(i);
772    });
773
774    payment31.setOnClickListener(view -> {
775        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
776        Bundle bundle31 = new Bundle();
777        bundle31.putString("norek", no31);
778        bundle31.putString("bank", banktxt31);
779        bundle31.putInt("price", price);
780        bundle31.putInt("amount", amount);
781
782        switch (type){
783            case 1:
784                bundle31.putSerializable("object", tix);
785                break;
786            case 2:
787                bundle31.putSerializable("object", htl);
788                break;
789            case 3:
790                bundle31.putSerializable("object", trp);
791                break;
792        }
793
794        i.putExtras(bundle31);
795        startActivity(i);
796    });
797
798    payment32.setOnClickListener(view -> {
799        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
800        Bundle bundle32 = new Bundle();
801        bundle32.putString("norek", no32);
802        bundle32.putString("bank", banktxt32);
803        bundle32.putInt("price", price);
804        bundle32.putInt("amount", amount);
805
806        switch (type){
807            case 1:
808                bundle32.putSerializable("object", tix);
809                break;
810            case 2:
811                bundle32.putSerializable("object", htl);
812                break;
813            case 3:
814                bundle32.putSerializable("object", trp);
815                break;
816        }
817
818        i.putExtras(bundle32);
819        startActivity(i);
820    });
821
822    payment33.setOnClickListener(view -> {
823        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
824        Bundle bundle33 = new Bundle();
825        bundle33.putString("norek", no33);
826        bundle33.putString("bank", banktxt33);
827        bundle33.putInt("price", price);
828        bundle33.putInt("amount", amount);
829
830        switch (type){
831            case 1:
832                bundle33.putSerializable("object", tix);
833                break;
834            case 2:
835                bundle33.putSerializable("object", htl);
836                break;
837            case 3:
838                bundle33.putSerializable("object", trp);
839                break;
840        }
841
842        i.putExtras(bundle33);
843        startActivity(i);
844    });
845
846    payment34.setOnClickListener(view -> {
847        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
848        Bundle bundle34 = new Bundle();
849        bundle34.putString("norek", no34);
850        bundle34.putString("bank", banktxt34);
851        bundle34.putInt("price", price);
852        bundle34.putInt("amount", amount);
853
854        switch (type){
855            case 1:
856                bundle34.putSerializable("object", tix);
857                break;
858            case 2:
859                bundle34.putSerializable("object", htl);
860                break;
861            case 3:
862                bundle34.putSerializable("object", trp);
863                break;
864        }
865
866        i.putExtras(bundle34);
867        startActivity(i);
868    });
869
870    payment35.setOnClickListener(view -> {
871        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
872        Bundle bundle35 = new Bundle();
873        bundle35.putString("norek", no35);
874        bundle35.putString("bank", banktxt35);
875        bundle35.putInt("price", price);
876        bundle35.putInt("amount", amount);
877
878        switch (type){
879            case 1:
880                bundle35.putSerializable("object", tix);
881                break;
882            case 2:
883                bundle35.putSerializable("object", htl);
884                break;
885            case 3:
886                bundle35.putSerializable("object", trp);
887                break;
888        }
889
890        i.putExtras(bundle35);
891        startActivity(i);
892    });
893
894    payment36.setOnClickListener(view -> {
895        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
896        Bundle bundle36 = new Bundle();
897        bundle36.putString("norek", no36);
898        bundle36.putString("bank", banktxt36);
899        bundle36.putInt("price", price);
900        bundle36.putInt("amount", amount);
901
902        switch (type){
903            case 1:
904                bundle36.putSerializable("object", tix);
905                break;
906            case 2:
907                bundle36.putSerializable("object", htl);
908                break;
909            case 3:
910                bundle36.putSerializable("object", trp);
911                break;
912        }
913
914        i.putExtras(bundle36);
915        startActivity(i);
916    });
917
918    payment37.setOnClickListener(view -> {
919        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
920        Bundle bundle37 = new Bundle();
921        bundle37.putString("norek", no37);
922        bundle37.putString("bank", banktxt37);
923        bundle37.putInt("price", price);
924        bundle37.putInt("amount", amount);
925
926        switch (type){
927            case 1:
928                bundle37.putSerializable("object", tix);
929                break;
930            case 2:
931                bundle37.putSerializable("object", htl);
932                break;
933            case 3:
934                bundle37.putSerializable("object", trp);
935                break;
936        }
937
938        i.putExtras(bundle37);
939        startActivity(i);
940    });
941
942    payment38.setOnClickListener(view -> {
943        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
944        Bundle bundle38 = new Bundle();
945        bundle38.putString("norek", no38);
946        bundle38.putString("bank", banktxt38);
947        bundle38.putInt("price", price);
948        bundle38.putInt("amount", amount);
949
950        switch (type){
951            case 1:
952                bundle38.putSerializable("object", tix);
953                break;
954            case 2:
955                bundle38.putSerializable("object", htl);
956                break;
957            case 3:
958                bundle38.putSerializable("object", trp);
959                break;
960        }
961
962        i.putExtras(bundle38);
963        startActivity(i);
964    });
965
966    payment39.setOnClickListener(view -> {
967        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
968        Bundle bundle39 = new Bundle();
969        bundle39.putString("norek", no39);
970        bundle39.putString("bank", banktxt39);
971        bundle39.putInt("price", price);
972        bundle39.putInt("amount", amount);
973
974        switch (type){
975            case 1:
976                bundle39.putSerializable("object", tix);
977                break;
978            case 2:
979                bundle39.putSerializable("object", htl);
980                break;
981            case 3:
982                bundle39.putSerializable("object", trp);
983                break;
984        }
985
986        i.putExtras(bundle39);
987        startActivity(i);
988    });
989
990    payment40.setOnClickListener(view -> {
991        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
992        Bundle bundle40 = new Bundle();
993        bundle40.putString("norek", no40);
994        bundle40.putString("bank", banktxt40);
995        bundle40.putInt("price", price);
996        bundle40.putInt("amount", amount);
997
998        switch (type){
999            case 1:
1000                bundle40.putSerializable("object", tix);
1001                break;
1002            case 2:
1003                bundle40.putSerializable("object", htl);
1004                break;
1005            case 3:
1006                bundle40.putSerializable("object", trp);
1007                break;
1008        }
1009
1010        i.putExtras(bundle40);
1011        startActivity(i);
1012    });
1013
1014    payment41.setOnClickListener(view -> {
1015        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
1016        Bundle bundle41 = new Bundle();
1017        bundle41.putString("norek", no41);
1018        bundle41.putString("bank", banktxt41);
1019        bundle41.putInt("price", price);
1020        bundle41.putInt("amount", amount);
1021
1022        switch (type){
1023            case 1:
1024                bundle41.putSerializable("object", tix);
1025                break;
1026            case 2:
1027                bundle41.putSerializable("object", htl);
1028                break;
1029            case 3:
1030                bundle41.putSerializable("object", trp);
1031                break;
1032        }
1033
1034        i.putExtras(bundle41);
1035        startActivity(i);
1036    });
1037
1038    payment42.setOnClickListener(view -> {
1039        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
1040        Bundle bundle42 = new Bundle();
1041        bundle42.putString("norek", no42);
1042        bundle42.putString("bank", banktxt42);
1043        bundle42.putInt("price", price);
1044        bundle42.putInt("amount", amount);
1045
1046        switch (type){
1047            case 1:
1048                bundle42.putSerializable("object", tix);
1049                break;
1050            case 2:
1051                bundle42.putSerializable("object", htl);
1052                break;
1053            case 3:
1054                bundle42.putSerializable("object", trp);
1055                break;
1056        }
1057
1058        i.putExtras(bundle42);
1059        startActivity(i);
1060    });
1061
1062    payment43.setOnClickListener(view -> {
1063        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
1064        Bundle bundle43 = new Bundle();
1065        bundle43.putString("norek", no43);
1066        bundle43.putString("bank", banktxt43);
1067        bundle43.putInt("price", price);
1068        bundle43.putInt("amount", amount);
1069
1070        switch (type){
1071            case 1:
1072                bundle43.putSerializable("object", tix);
1073                break;
1074            case 2:
1075                bundle43.putSerializable("object", htl);
1076                break;
1077            case 3:
1078                bundle43.putSerializable("object", trp);
1079                break;
1080        }
1081
1082        i.putExtras(bundle43);
1083        startActivity(i);
1084    });
1085
1086    payment44.setOnClickListener(view -> {
1087        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
1088        Bundle bundle44 = new Bundle();
1089        bundle44.putString("norek", no44);
1090        bundle44.putString("bank", banktxt44);
1091        bundle44.putInt("price", price);
1092        bundle44.putInt("amount", amount);
1093
1094        switch (type){
1095            case 1:
1096                bundle44.putSerializable("object", tix);
1097                break;
1098            case 2:
1099                bundle44.putSerializable("object", htl);
1100                break;
1101            case 3:
1102                bundle44.putSerializable("object", trp);
1103                break;
1104        }
1105
1106        i.putExtras(bundle44);
1107        startActivity(i);
1108    });
1109
1110    payment45.setOnClickListener(view -> {
1111        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
1112        Bundle bundle45 = new Bundle();
1113        bundle45.putString("norek", no45);
1114        bundle45.putString("bank", banktxt45);
1115        bundle45.putInt("price", price);
1116        bundle45.putInt("amount", amount);
1117
1118        switch (type){
1119            case 1:
1120                bundle45.putSerializable("object", tix);
1121                break;
1122            case 2:
1123                bundle45.putSerializable("object", htl);
1124                break;
1125            case 3:
1126                bundle45.putSerializable("object", trp);
1127                break;
1128        }
1129
1130        i.putExtras(bundle45);
1131        startActivity(i);
1132    });
1133
1134    payment46.setOnClickListener(view -> {
1135        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
1136        Bundle bundle46 = new Bundle();
1137        bundle46.putString("norek", no46);
1138        bundle46.putString("bank", banktxt46);
1139        bundle46.putInt("price", price);
1140        bundle46.putInt("amount", amount);
1141
1142        switch (type){
1143            case 1:
1144                bundle46.putSerializable("object", tix);
1145                break;
1146            case 2:
1147                bundle46.putSerializable("object", htl);
1148                break;
1149            case 3:
1150                bundle46.putSerializable("object", trp);
1151                break;
1152        }
1153
1154        i.putExtras(bundle46);
1155        startActivity(i);
1156    });
1157
1158    payment47.setOnClickListener(view -> {
1159        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
1160        Bundle bundle47 = new Bundle();
1161        bundle47.putString("norek", no47);
1162        bundle47.putString("bank", banktxt47);
1163        bundle47.putInt("price", price);
1164        bundle47.putInt("amount", amount);
1165
1166        switch (type){
1167            case 1:
1168                bundle47.putSerializable("object", tix);
1169                break;
1170            case 2:
1171                bundle47.putSerializable("object", htl);
1172                break;
1173            case 3:
1174                bundle47.putSerializable("object", trp);
1175                break;
1176        }
1177
1178        i.putExtras(bundle47);
1179        startActivity(i);
1180    });
1181
1182    payment48.setOnClickListener(view -> {
1183        Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
1184        Bundle bundle48 = new Bundle();
1185        bundle48.putString("norek
```

```
PaymentActivity.java
101     bundle2.putInt("draw", R.drawable.mandiri);
102     i.putExtras(bundle2);
103     startActivity(i);
104 };
105 payment3.setOnClickListener(view -> {
106     Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
107     Bundle bundle3 = new Bundle();
108     bundle3.putString("norek", no3);
109     bundle3.putInt("price", price);
110     bundle3.putInt("amount", amount);
111     bundle3.putString("bank", banktxt3);
112
113     switch (type){
114         case 1:
115             bundle3.putSerializable("object", tix);
116             break;
117         case 2:
118             bundle3.putSerializable("object", htl);
119             break;
120         case 3:
121             bundle3.putSerializable("object", trp);
122             break;
123     }
124
125     bundle3.putInt("draw", R.drawable.bni);
126     i.putExtras(bundle3);
127     startActivity(i);
128 };
129 payment4.setOnClickListener(view -> {
130     Intent i = new Intent(getApplicationContext(), PaymentDetailActivity.class);
131     Bundle bundle4 = new Bundle();
```

```
PaymentActivity.java
132     bundle4.putString("norek", no4);
133     bundle4.putInt("price", price);
134     bundle4.putInt("amount", amount);
135     bundle4.putString("bank", banktxt4);
136
137     switch (type){
138         case 1:
139             bundle4.putSerializable("object", tix);
140             break;
141         case 2:
142             bundle4.putSerializable("object", htl);
143             break;
144         case 3:
145             bundle4.putSerializable("object", trp);
146             break;
147     }
148
149     bundle4.putInt("draw", R.drawable.bri);
150     i.putExtras(bundle4);
151     startActivity(i);
152 };
153 }
```

Kode java PaymentActivity di atas merupakan implementasi dari *PaymentActivity* dalam aplikasi Android yang digunakan untuk mengelola proses pembayaran untuk berbagai jenis produk seperti tiket, hotel, dan tour. Aktivitas ini menginisialisasi elemen UI seperti *TextView* untuk tombol pembayaran dan *FrameLayout* untuk tombol kembali. Data harga dan jumlah produk diambil dari Intent yang memulai aktivitas ini, dan objek produk yang dipilih (tiket, hotel, atau tour) juga diambil dan diidentifikasi. Setiap tombol pembayaran (*payment1*, *payment2*, *payment3*, *payment4*) memiliki *listener* yang mengarahkan pengguna ke *PaymentDetailActivity* dengan membawa informasi yang diperlukan seperti nomor rekening bank, nama bank, harga, jumlah, dan jenis produk yang dipilih. Informasi ini dikemas dalam *Bundle* dan ditambahkan ke *Intent* sebelum memulai aktivitas baru. Tombol kembali memungkinkan pengguna untuk kembali ke layar sebelumnya. Kode ini mengimplementasikan fitur untuk memilih metode pembayaran dan menampilkan detail pembayaran berdasarkan produk yang dipilih oleh pengguna.

### 3.2.33. *PaymentDetailActivity*



```

1  package com.example.project;
2
3  import ...
4
5  public class PaymentDetailActivity extends AppCompatActivity {
6      private final int duration = 360;
7      TextView txtRek, txtAmount, txtBank;
8      ImageView imgBank;
9      LinearLayout pb;
10     int img, type, amount, price;
11     String noRek, bank;
12     ticket tix;
13     hotel htl;
14     tour trp;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.activity_payment_detail);
20         txtRek = findViewById(R.id.txtNoRek);
21         txtBank = findViewById(R.id.txtBank);
22         txtAmount = findViewById(R.id.txtAmount);
23         imgBank = findViewById(R.id.imgBank);
24         pb = findViewById(R.id.progressBarPayment);
25
26         pb.setVisibility(LinearLayout.INVISIBLE);
27
28         MaterialButton copyButtonRek = findViewById(R.id.btnCopyRek);
29         MaterialButton copyButtonAmount = findViewById(R.id.btnCopyAmount);
30         MaterialButton btnDone = findViewById(R.id.btnDone);
31
32     }
33
34     public void payment1(View view) {
35         Intent intent = new Intent(this, PaymentActivity.class);
36         intent.putExtra("type", "ticket");
37         intent.putExtra("amount", amount);
38         intent.putExtra("img", img);
39         intent.putExtra("noRek", noRek);
40         intent.putExtra("bank", bank);
41         intent.putExtra("tix", tix);
42         intent.putExtra("htl", htl);
43         intent.putExtra("trp", trp);
44         intent.putExtra("duration", duration);
45         intent.putExtra("pb", pb);
46         intent.putExtra("imgBank", imgBank);
47         intent.putExtra("copyButtonRek", copyButtonRek);
48         intent.putExtra("copyButtonAmount", copyButtonAmount);
49         intent.putExtra("btnDone", btnDone);
50
51         startActivity(intent);
52     }
53
54     public void payment2(View view) {
55         Intent intent = new Intent(this, PaymentActivity.class);
56         intent.putExtra("type", "hotel");
57         intent.putExtra("amount", amount);
58         intent.putExtra("img", img);
59         intent.putExtra("noRek", noRek);
60         intent.putExtra("bank", bank);
61         intent.putExtra("tix", tix);
62         intent.putExtra("htl", htl);
63         intent.putExtra("trp", trp);
64         intent.putExtra("duration", duration);
65         intent.putExtra("pb", pb);
66         intent.putExtra("imgBank", imgBank);
67         intent.putExtra("copyButtonRek", copyButtonRek);
68         intent.putExtra("copyButtonAmount", copyButtonAmount);
69         intent.putExtra("btnDone", btnDone);
70
71         startActivity(intent);
72     }
73
74     public void payment3(View view) {
75         Intent intent = new Intent(this, PaymentActivity.class);
76         intent.putExtra("type", "tour");
77         intent.putExtra("amount", amount);
78         intent.putExtra("img", img);
79         intent.putExtra("noRek", noRek);
80         intent.putExtra("bank", bank);
81         intent.putExtra("tix", tix);
82         intent.putExtra("htl", htl);
83         intent.putExtra("trp", trp);
84         intent.putExtra("duration", duration);
85         intent.putExtra("pb", pb);
86         intent.putExtra("imgBank", imgBank);
87         intent.putExtra("copyButtonRek", copyButtonRek);
88         intent.putExtra("copyButtonAmount", copyButtonAmount);
89         intent.putExtra("btnDone", btnDone);
90
91         startActivity(intent);
92     }
93
94     public void payment4(View view) {
95         Intent intent = new Intent(this, PaymentActivity.class);
96         intent.putExtra("type", "ticket");
97         intent.putExtra("amount", amount);
98         intent.putExtra("img", img);
99         intent.putExtra("noRek", noRek);
100        intent.putExtra("bank", bank);
101        intent.putExtra("tix", tix);
102        intent.putExtra("htl", htl);
103        intent.putExtra("trp", trp);
104        intent.putExtra("duration", duration);
105        intent.putExtra("pb", pb);
106        intent.putExtra("imgBank", imgBank);
107        intent.putExtra("copyButtonRek", copyButtonRek);
108        intent.putExtra("copyButtonAmount", copyButtonAmount);
109        intent.putExtra("btnDone", btnDone);
110
111        startActivity(intent);
112    }
113
114 }

```

```
73         Bundle b = getIntent().getExtras();
74         noRek = b.getString( key: "norek");
75         bank = b.getString( key: "bank");
76         img = b.getInt( key: "draw");
77         price = b.getInt( key: "price");
78         amount = b.getInt( key: "amount");
79
80         if(b.getSerializable( key: "object") instanceof ticket){
81             tix = (ticket) b.getSerializable( key: "object");
82             type = 1;
83         }else if(b.getSerializable( key: "object") instanceof hotel){
84             htl = (hotel) b.getSerializable( key: "object");
85             type = 2;
86         }else if(b.getSerializable( key: "object") instanceof tour){
87             trp = (tour) b.getSerializable( key: "object");
88             type = 3;
89         }
90
91         txtBank.setText(bank);
92         txtRek.setText(noRek);
93         txtAmount.setText(String.valueOf(price));
94         imgBank.setImageResource(img);
95
96         new CountDownTimer( millisInFuture: duration * 1000L, countDownInterval: 1000) {
97             @Override
98             public void onTick(long millisUntilFinished) {
99                 long totalMinutes = TimeUnit.MILLISECONDS.toMinutes(millisUntilFinished);
100                long totalSeconds = TimeUnit.MILLISECONDS.toSeconds(millisUntilFinished);
101
102                long hours = totalMinutes / 60;
103                long minutes = totalMinutes % 60;
104
105                String hoursString = String.valueOf(hours);
106                String minutesString = String.valueOf(minutes);
107
108                if(hours < 10) {
109                    hoursString = "0" + hoursString;
110                }
111
112                if(minutes < 10) {
113                    minutesString = "0" + minutesString;
114                }
115
116                String timeString = hoursString + ":" + minutesString;
117
118                txtTime.setText(timeString);
119            }
120        }.start();
121    }
122
123    @Override
124    public void onBackPressed() {
125        Intent intent = new Intent(this, MainActivity.class);
126        startActivity(intent);
127        finish();
128    }
129
130    @Override
131    protected void onDestroy() {
132        super.onDestroy();
133        if(timer != null) {
134            timer.cancel();
135        }
136    }
137
138    @Override
139    protected void onPause() {
140        super.onPause();
141        if(timer != null) {
142            timer.cancel();
143        }
144    }
145
146    @Override
147    protected void onResume() {
148        super.onResume();
149        if(timer != null) {
150            timer.start();
151        }
152    }
153
154    @Override
155    protected void onStart() {
156        super.onStart();
157        if(timer != null) {
158            timer.start();
159        }
160    }
161
162    @Override
163    protected void onStop() {
164        super.onStop();
165        if(timer != null) {
166            timer.cancel();
167        }
168    }
169
170    @Override
171    protected void onRestart() {
172        super.onRestart();
173        if(timer != null) {
174            timer.start();
175        }
176    }
177
178    @Override
179    protected void onConfigurationChanged(Configuration newConfig) {
180        super.onConfigurationChanged(newConfig);
181        if(timer != null) {
182            timer.start();
183        }
184    }
185
186    @Override
187    protected void onLowMemory() {
188        super.onLowMemory();
189        if(timer != null) {
190            timer.cancel();
191        }
192    }
193
194    @Override
195    protected void onTrimMemory(int level) {
196        super.onTrimMemory(level);
197        if(timer != null) {
198            timer.cancel();
199        }
200    }
201
202    @Override
203    protected void onNewIntent(Intent intent) {
204        super.onNewIntent(intent);
205        if(timer != null) {
206            timer.cancel();
207        }
208    }
209
210    @Override
211    protected void onTrimMemory(int level) {
212        super.onTrimMemory(level);
213        if(timer != null) {
214            timer.cancel();
215        }
216    }
217
218    @Override
219    protected void onNewIntent(Intent intent) {
220        super.onNewIntent(intent);
221        if(timer != null) {
222            timer.cancel();
223        }
224    }
225
226    @Override
227    protected void onTrimMemory(int level) {
228        super.onTrimMemory(level);
229        if(timer != null) {
230            timer.cancel();
231        }
232    }
233
234    @Override
235    protected void onNewIntent(Intent intent) {
236        super.onNewIntent(intent);
237        if(timer != null) {
238            timer.cancel();
239        }
240    }
241
242    @Override
243    protected void onLowMemory() {
244        super.onLowMemory();
245        if(timer != null) {
246            timer.cancel();
247        }
248    }
249
250    @Override
251    protected void onConfigurationChanged(Configuration newConfig) {
252        super.onConfigurationChanged(newConfig);
253        if(timer != null) {
254            timer.start();
255        }
256    }
257
258    @Override
259    protected void onTrimMemory(int level) {
260        super.onTrimMemory(level);
261        if(timer != null) {
262            timer.cancel();
263        }
264    }
265
266    @Override
267    protected void onNewIntent(Intent intent) {
268        super.onNewIntent(intent);
269        if(timer != null) {
270            timer.cancel();
271        }
272    }
273
274    @Override
275    protected void onLowMemory() {
276        super.onLowMemory();
277        if(timer != null) {
278            timer.cancel();
279        }
280    }
281
282    @Override
283    protected void onConfigurationChanged(Configuration newConfig) {
284        super.onConfigurationChanged(newConfig);
285        if(timer != null) {
286            timer.start();
287        }
288    }
289
290    @Override
291    protected void onTrimMemory(int level) {
292        super.onTrimMemory(level);
293        if(timer != null) {
294            timer.cancel();
295        }
296    }
297
298    @Override
299    protected void onNewIntent(Intent intent) {
300        super.onNewIntent(intent);
301        if(timer != null) {
302            timer.cancel();
303        }
304    }
305
306    @Override
307    protected void onLowMemory() {
308        super.onLowMemory();
309        if(timer != null) {
310            timer.cancel();
311        }
312    }
313
314    @Override
315    protected void onConfigurationChanged(Configuration newConfig) {
316        super.onConfigurationChanged(newConfig);
317        if(timer != null) {
318            timer.start();
319        }
320    }
321
322    @Override
323    protected void onTrimMemory(int level) {
324        super.onTrimMemory(level);
325        if(timer != null) {
326            timer.cancel();
327        }
328    }
329
330    @Override
331    protected void onNewIntent(Intent intent) {
332        super.onNewIntent(intent);
333        if(timer != null) {
334            timer.cancel();
335        }
336    }
337
338    @Override
339    protected void onLowMemory() {
340        super.onLowMemory();
341        if(timer != null) {
342            timer.cancel();
343        }
344    }
345
346    @Override
347    protected void onConfigurationChanged(Configuration newConfig) {
348        super.onConfigurationChanged(newConfig);
349        if(timer != null) {
350            timer.start();
351        }
352    }
353
354    @Override
355    protected void onTrimMemory(int level) {
356        super.onTrimMemory(level);
357        if(timer != null) {
358            timer.cancel();
359        }
360    }
361
362    @Override
363    protected void onNewIntent(Intent intent) {
364        super.onNewIntent(intent);
365        if(timer != null) {
366            timer.cancel();
367        }
368    }
369
370    @Override
371    protected void onLowMemory() {
372        super.onLowMemory();
373        if(timer != null) {
374            timer.cancel();
375        }
376    }
377
378    @Override
379    protected void onConfigurationChanged(Configuration newConfig) {
380        super.onConfigurationChanged(newConfig);
381        if(timer != null) {
382            timer.start();
383        }
384    }
385
386    @Override
387    protected void onTrimMemory(int level) {
388        super.onTrimMemory(level);
389        if(timer != null) {
390            timer.cancel();
391        }
392    }
393
394    @Override
395    protected void onNewIntent(Intent intent) {
396        super.onNewIntent(intent);
397        if(timer != null) {
398            timer.cancel();
399        }
400    }
401
402    @Override
403    protected void onLowMemory() {
404        super.onLowMemory();
405        if(timer != null) {
406            timer.cancel();
407        }
408    }
409
410    @Override
411    protected void onConfigurationChanged(Configuration newConfig) {
412        super.onConfigurationChanged(newConfig);
413        if(timer != null) {
414            timer.start();
415        }
416    }
417
418    @Override
419    protected void onTrimMemory(int level) {
420        super.onTrimMemory(level);
421        if(timer != null) {
422            timer.cancel();
423        }
424    }
425
426    @Override
427    protected void onNewIntent(Intent intent) {
428        super.onNewIntent(intent);
429        if(timer != null) {
430            timer.cancel();
431        }
432    }
433
434    @Override
435    protected void onLowMemory() {
436        super.onLowMemory();
437        if(timer != null) {
438            timer.cancel();
439        }
440    }
441
442    @Override
443    protected void onConfigurationChanged(Configuration newConfig) {
444        super.onConfigurationChanged(newConfig);
445        if(timer != null) {
446            timer.start();
447        }
448    }
449
450    @Override
451    protected void onTrimMemory(int level) {
452        super.onTrimMemory(level);
453        if(timer != null) {
454            timer.cancel();
455        }
456    }
457
458    @Override
459    protected void onNewIntent(Intent intent) {
460        super.onNewIntent(intent);
461        if(timer != null) {
462            timer.cancel();
463        }
464    }
465
466    @Override
467    protected void onLowMemory() {
468        super.onLowMemory();
469        if(timer != null) {
470            timer.cancel();
471        }
472    }
473
474    @Override
475    protected void onConfigurationChanged(Configuration newConfig) {
476        super.onConfigurationChanged(newConfig);
477        if(timer != null) {
478            timer.start();
479        }
480    }
481
482    @Override
483    protected void onTrimMemory(int level) {
484        super.onTrimMemory(level);
485        if(timer != null) {
486            timer.cancel();
487        }
488    }
489
490    @Override
491    protected void onNewIntent(Intent intent) {
492        super.onNewIntent(intent);
493        if(timer != null) {
494            timer.cancel();
495        }
496    }
497
498    @Override
499    protected void onLowMemory() {
500        super.onLowMemory();
501        if(timer != null) {
502            timer.cancel();
503        }
504    }
505
506    @Override
507    protected void onConfigurationChanged(Configuration newConfig) {
508        super.onConfigurationChanged(newConfig);
509        if(timer != null) {
510            timer.start();
511        }
512    }
513
514    @Override
515    protected void onTrimMemory(int level) {
516        super.onTrimMemory(level);
517        if(timer != null) {
518            timer.cancel();
519        }
520    }
521
522    @Override
523    protected void onNewIntent(Intent intent) {
524        super.onNewIntent(intent);
525        if(timer != null) {
526            timer.cancel();
527        }
528    }
529
530    @Override
531    protected void onLowMemory() {
532        super.onLowMemory();
533        if(timer != null) {
534            timer.cancel();
535        }
536    }
537
538    @Override
539    protected void onConfigurationChanged(Configuration newConfig) {
540        super.onConfigurationChanged(newConfig);
541        if(timer != null) {
542            timer.start();
543        }
544    }
545
546    @Override
547    protected void onTrimMemory(int level) {
548        super.onTrimMemory(level);
549        if(timer != null) {
550            timer.cancel();
551        }
552    }
553
554    @Override
555    protected void onNewIntent(Intent intent) {
556        super.onNewIntent(intent);
557        if(timer != null) {
558            timer.cancel();
559        }
560    }
561
562    @Override
563    protected void onLowMemory() {
564        super.onLowMemory();
565        if(timer != null) {
566            timer.cancel();
567        }
568    }
569
570    @Override
571    protected void onConfigurationChanged(Configuration newConfig) {
572        super.onConfigurationChanged(newConfig);
573        if(timer != null) {
574            timer.start();
575        }
576    }
577
578    @Override
579    protected void onTrimMemory(int level) {
580        super.onTrimMemory(level);
581        if(timer != null) {
582            timer.cancel();
583        }
584    }
585
586    @Override
587    protected void onNewIntent(Intent intent) {
588        super.onNewIntent(intent);
589        if(timer != null) {
590            timer.cancel();
591        }
592    }
593
594    @Override
595    protected void onLowMemory() {
596        super.onLowMemory();
597        if(timer != null) {
598            timer.cancel();
599        }
600    }
601
602    @Override
603    protected void onConfigurationChanged(Configuration newConfig) {
604        super.onConfigurationChanged(newConfig);
605        if(timer != null) {
606            timer.start();
607        }
608    }
609
610    @Override
611    protected void onTrimMemory(int level) {
612        super.onTrimMemory(level);
613        if(timer != null) {
614            timer.cancel();
615        }
616    }
617
618    @Override
619    protected void onNewIntent(Intent intent) {
620        super.onNewIntent(intent);
621        if(timer != null) {
622            timer.cancel();
623        }
624    }
625
626    @Override
627    protected void onLowMemory() {
628        super.onLowMemory();
629        if(timer != null) {
630            timer.cancel();
631        }
632    }
633
634    @Override
635    protected void onConfigurationChanged(Configuration newConfig) {
636        super.onConfigurationChanged(newConfig);
637        if(timer != null) {
638            timer.start();
639        }
640    }
641
642    @Override
643    protected void onTrimMemory(int level) {
644        super.onTrimMemory(level);
645        if(timer != null) {
646            timer.cancel();
647        }
648    }
649
650    @Override
651    protected void onNewIntent(Intent intent) {
652        super.onNewIntent(intent);
653        if(timer != null) {
654            timer.cancel();
655        }
656    }
657
658    @Override
659    protected void onLowMemory() {
660        super.onLowMemory();
661        if(timer != null) {
662            timer.cancel();
663        }
664    }
665
666    @Override
667    protected void onConfigurationChanged(Configuration newConfig) {
668        super.onConfigurationChanged(newConfig);
669        if(timer != null) {
670            timer.start();
671        }
672    }
673
674    @Override
675    protected void onTrimMemory(int level) {
676        super.onTrimMemory(level);
677        if(timer != null) {
678            timer.cancel();
679        }
680    }
681
682    @Override
683    protected void onNewIntent(Intent intent) {
684        super.onNewIntent(intent);
685        if(timer != null) {
686            timer.cancel();
687        }
688    }
689
690    @Override
691    protected void onLowMemory() {
692        super.onLowMemory();
693        if(timer != null) {
694            timer.cancel();
695        }
696    }
697
698    @Override
699    protected void onConfigurationChanged(Configuration newConfig) {
700        super.onConfigurationChanged(newConfig);
701        if(timer != null) {
702            timer.start();
703        }
704    }
705
706    @Override
707    protected void onTrimMemory(int level) {
708        super.onTrimMemory(level);
709        if(timer != null) {
710            timer.cancel();
711        }
712    }
713
714    @Override
715    protected void onNewIntent(Intent intent) {
716        super.onNewIntent(intent);
717        if(timer != null) {
718            timer.cancel();
719        }
720    }
721
722    @Override
723    protected void onLowMemory() {
724        super.onLowMemory();
725        if(timer != null) {
726            timer.cancel();
727        }
728    }
729
730    @Override
731    protected void onConfigurationChanged(Configuration newConfig) {
732        super.onConfigurationChanged(newConfig);
733        if(timer != null) {
734            timer.start();
735        }
736    }
737
738    @Override
739    protected void onTrimMemory(int level) {
740        super.onTrimMemory(level);
741        if(timer != null) {
742            timer.cancel();
743        }
744    }
745
746    @Override
747    protected void onNewIntent(Intent intent) {
748        super.onNewIntent(intent);
749        if(timer != null) {
750            timer.cancel();
751        }
752    }
753
754    @Override
755    protected void onLowMemory() {
756        super.onLowMemory();
757        if(timer != null) {
758            timer.cancel();
759        }
760    }
761
762    @Override
763    protected void onConfigurationChanged(Configuration newConfig) {
764        super.onConfigurationChanged(newConfig);
765        if(timer != null) {
766            timer.start();
767        }
768    }
769
770    @Override
771    protected void onTrimMemory(int level) {
772        super.onTrimMemory(level);
773        if(timer != null) {
774            timer.cancel();
775        }
776    }
777
778    @Override
779    protected void onNewIntent(Intent intent) {
780        super.onNewIntent(intent);
781        if(timer != null) {
782            timer.cancel();
783        }
784    }
785
786    @Override
787    protected void onLowMemory() {
788        super.onLowMemory();
789        if(timer != null) {
790            timer.cancel();
791        }
792    }
793
794    @Override
795    protected void onConfigurationChanged(Configuration newConfig) {
796        super.onConfigurationChanged(newConfig);
797        if(timer != null) {
798            timer.start();
799        }
800    }
801
802    @Override
803    protected void onTrimMemory(int level) {
804        super.onTrimMemory(level);
805        if(timer != null) {
806            timer.cancel();
807        }
808    }
809
810    @Override
811    protected void onNewIntent(Intent intent) {
812        super.onNewIntent(intent);
813        if(timer != null) {
814            timer.cancel();
815        }
816    }
817
818    @Override
819    protected void onLowMemory() {
820        super.onLowMemory();
821        if(timer != null) {
822            timer.cancel();
823        }
824    }
825
826    @Override
827    protected void onConfigurationChanged(Configuration newConfig) {
828        super.onConfigurationChanged(newConfig);
829        if(timer != null) {
830            timer.start();
831        }
832    }
833
834    @Override
835    protected void onTrimMemory(int level) {
836        super.onTrimMemory(level);
837        if(timer != null) {
838            timer.cancel();
839        }
840    }
841
842    @Override
843    protected void onNewIntent(Intent intent) {
844        super.onNewIntent(intent);
845        if(timer != null) {
846            timer.cancel();
847        }
848    }
849
850    @Override
851    protected void onLowMemory() {
852        super.onLowMemory();
853        if(timer != null) {
854            timer.cancel();
855        }
856    }
857
858    @Override
859    protected void onConfigurationChanged(Configuration newConfig) {
860        super.onConfigurationChanged(newConfig);
861        if(timer != null) {
862            timer.start();
863        }
864    }
865
866    @Override
867    protected void onTrimMemory(int level) {
868        super.onTrimMemory(level);
869        if(timer != null) {
870            timer.cancel();
871        }
872    }
873
874    @Override
875    protected void onNewIntent(Intent intent) {
876        super.onNewIntent(intent);
877        if(timer != null) {
878            timer.cancel();
879        }
880    }
881
882    @Override
883    protected void onLowMemory() {
884        super.onLowMemory();
885        if(timer != null) {
886            timer.cancel();
887        }
888    }
889
890    @Override
891    protected void onConfigurationChanged(Configuration newConfig) {
892        super.onConfigurationChanged(newConfig);
893        if(timer != null) {
894            timer.start();
895        }
896    }
897
898    @Override
899    protected void onTrimMemory(int level) {
900        super.onTrimMemory(level);
901        if(timer != null) {
902            timer.cancel();
903        }
904    }
905
906    @Override
907    protected void onNewIntent(Intent intent) {
908        super.onNewIntent(intent);
909        if(timer != null) {
910            timer.cancel();
911        }
912    }
913
914    @Override
915    protected void onLowMemory() {
916        super.onLowMemory();
917        if(timer != null) {
918            timer.cancel();
919        }
920    }
921
922    @Override
923    protected void onConfigurationChanged(Configuration newConfig) {
924        super.onConfigurationChanged(newConfig);
925        if(timer != null) {
926            timer.start();
927        }
928    }
929
930    @Override
931    protected void onTrimMemory(int level) {
932        super.onTrimMemory(level);
933        if(timer != null) {
934            timer.cancel();
935        }
936    }
937
938    @Override
939    protected void onNewIntent(Intent intent) {
940        super.onNewIntent(intent);
941        if(timer != null) {
942            timer.cancel();
943        }
944    }
945
946    @Override
947    protected void onLowMemory() {
948        super.onLowMemory();
949        if(timer != null) {
950            timer.cancel();
951        }
952    }
953
954    @Override
955    protected void onConfigurationChanged(Configuration newConfig) {
956        super.onConfigurationChanged(newConfig);
957        if(timer != null) {
958            timer.start();
959        }
960    }
961
962    @Override
963    protected void onTrimMemory(int level) {
964        super.onTrimMemory(level);
965        if(timer != null) {
966            timer.cancel();
967        }
968    }
969
970    @Override
971    protected void onNewIntent(Intent intent) {
972        super.onNewIntent(intent);
973        if(timer != null) {
974            timer.cancel();
975        }
976    }
977
978    @Override
979    protected void onLowMemory() {
980        super.onLowMemory();
981        if(timer != null) {
982            timer.cancel();
983        }
984    }
985
986    @Override
987    protected void onConfigurationChanged(Configuration newConfig) {
988        super.onConfigurationChanged(newConfig);
989        if(timer != null) {
990            timer.start();
991        }
992    }
993
994    @Override
995    protected void onTrimMemory(int level) {
996        super.onTrimMemory(level);
997        if(timer != null) {
998            timer.cancel();
999        }
1000
1001    @Override
1002    protected void onNewIntent(Intent intent) {
1003        super.onNewIntent(intent);
1004        if(timer != null) {
1005            timer.cancel();
1006        }
1007    }
1008
1009    @Override
1010    protected void onLowMemory() {
1011        super.onLowMemory();
1012        if(timer != null) {
1013            timer.cancel();
1014        }
1015    }
1016
1017    @Override
1018    protected void onConfigurationChanged(Configuration newConfig) {
1019        super.onConfigurationChanged(newConfig);
1020        if(timer != null) {
1021            timer.start();
1022        }
1023    }
1024
1025    @Override
1026    protected void onTrimMemory(int level) {
1027        super.onTrimMemory(level);
1028        if(timer != null) {
1029            timer.cancel();
1030        }
1031    }
1032
1033    @Override
1034    protected void onNewIntent(Intent intent) {
1035        super.onNewIntent(intent);
1036        if(timer != null) {
1037            timer.cancel();
1038        }
1039    }
1040
1041    @Override
1042    protected void onLowMemory() {
1043        super.onLowMemory();
1044        if(timer != null) {
1045            timer.cancel();
1046        }
1047    }
1048
1049    @Override
1050    protected void onConfigurationChanged(Configuration newConfig) {
1051        super.onConfigurationChanged(newConfig);
1052        if(timer != null) {
1053            timer.start();
1054        }
1055    }
1056
1057    @Override
1058    protected void onTrimMemory(int level) {
1059        super.onTrimMemory(level);
1060        if(timer != null) {
1061            timer.cancel();
1062        }
1063    }
1064
1065    @Override
1066    protected void onNewIntent(Intent intent) {
1067        super.onNewIntent(intent);
1068        if(timer != null) {
1069            timer.cancel();
1070        }
1071    }
1072
1073    @Override
1074    protected void onLowMemory() {
1075        super.onLowMemory();
1076        if(timer != null) {
1077            timer.cancel();
1078        }
1079    }
1080
1081    @Override
1082    protected void onConfigurationChanged(Configuration newConfig) {
1083        super.onConfigurationChanged(newConfig);
1084        if(timer != null) {
1085            timer.start();
1086        }
1087    }
1088
1089    @Override
1090    protected void onTrimMemory(int level) {
1091        super.onTrimMemory(level);
1092        if(timer != null) {
1093            timer.cancel();
1094        }
1095    }
1096
1097    @Override
1098    protected void onNewIntent(Intent intent) {
1099        super.onNewIntent(intent);
1100        if(timer != null) {
1101            timer.cancel();
1102        }
1103    }
1104
1105    @Override
1106    protected void onLowMemory() {
1107        super.onLowMemory();
1108        if(timer != null) {
1109            timer.cancel();
1110        }
1111    }
1112
1113    @Override
1114    protected void onConfigurationChanged(Configuration newConfig) {
1115        super.onConfigurationChanged(newConfig);
1116        if(timer != null) {
1117            timer.start();
1118        }
1119    }
1120
1121    @Override
1122    protected void onTrimMemory(int level) {
1123        super.onTrimMemory(level);
1124        if(timer != null) {
1125            timer.cancel();
1126        }
1127    }
1128
1129    @Override
1130    protected void onNewIntent(Intent intent) {
1131        super.onNewIntent(intent);
1132        if(timer != null) {
1133            timer.cancel();
1134        }
1135    }
1136
1137    @Override
1138    protected void onLowMemory() {
1139        super.onLowMemory();
1140        if(timer != null) {
1141            timer.cancel();
1142        }
1143    }
1144
1145    @Override
1146    protected void onConfigurationChanged(Configuration newConfig) {
1147        super.onConfigurationChanged(newConfig);
1148        if(timer != null) {
1149            timer.start();
1150        }
1151    }
1152
1153    @Override
1154    protected void onTrimMemory(int level) {
1155        super.onTrimMemory(level);
1156        if(timer != null) {
1157            timer.cancel();
1158        }
1159    }
1160
1161    @Override
1162    protected void onNewIntent(Intent intent) {
1163        super.onNewIntent(intent);
1164        if(timer != null) {
1165            timer.cancel();
1166        }
1167    }
1168
1169    @Override
1170    protected void onLowMemory() {
1171        super.onLowMemory();
1172        if(timer != null) {
1173            timer.cancel();
1174        }
1175    }
1176
1177    @Override
1178    protected void onConfigurationChanged(Configuration newConfig) {
1179        super.onConfigurationChanged(newConfig);
1180        if(timer != null) {
1181            timer.start();
1182        }
1183    }
1184
1185    @Override
1186    protected void onTrimMemory(int level) {
1187        super.onTrimMemory(level);
1188        if(timer != null) {
1189            timer.cancel();
1190        }
1191    }
1192
1193    @Override
1194    protected void onNewIntent(Intent intent) {
1195        super.onNewIntent(intent);
1196        if(timer != null) {
1197            timer.cancel();
1198        }
1199    }
1200
1201    @Override
1202    protected void onLowMemory() {
1203        super.onLowMemory();
1204        if(timer != null) {
1205            timer.cancel();
1206        }
1207    }
1208
1209    @Override
1210    protected void onConfigurationChanged(Configuration newConfig) {
1211        super.onConfigurationChanged(newConfig);
1212        if(timer != null) {
1213            timer.start();
1214        }
1215    }
1216
1217    @Override
1218    protected void onTrimMemory(int level) {
1219        super.onTrimMemory(level);
1220        if(timer != null) {
1221            timer.cancel();
1222        }
1223    }
1224
1225    @Override
1226    protected void onNewIntent(Intent intent) {
1227        super.onNewIntent(intent);
1228        if(timer != null) {
1229            timer.cancel();
1230        }
1231    }
1232
1233    @Override
1234    protected void onLowMemory() {
1235        super.onLowMemory();
1236        if(timer != null) {
1237            timer.cancel();
1238        }
1239    }
1240
1241    @Override
1242    protected void onConfigurationChanged(Configuration newConfig) {
1243        super.onConfigurationChanged(newConfig);
1244        if(timer != null) {
1245            timer.start();
1246        }
1247    }
1248
1249    @Override
1250    protected void onTrimMemory(int level) {
1251        super.onTrimMemory(level);
1252        if(timer != null) {
1253            timer.cancel();
1254        }
1255    }
1256
1257    @Override
1258    protected void onNewIntent(Intent intent) {
1259        super.onNewIntent(intent);
1260        if(timer != null) {
1261            timer.cancel();
1262        }
1263    }
1264
1265    @Override
1266    protected void onLowMemory() {
1267        super.onLowMemory();
1268        if(timer != null) {
1269            timer.cancel();
1270        }
1271    }
1272
1273    @Override
1274    protected void onConfigurationChanged(Configuration newConfig) {
1275        super.onConfigurationChanged(newConfig);
1276        if(timer != null) {
1277            timer.start();
1278        }
1279    }
1280
1281    @Override
1282    protected void onTrimMemory(int level) {
1283        super.onTrimMemory(level);
1284        if(timer != null) {
1285            timer.cancel();
1286        }
1287    }
1288
1289    @Override
1290    protected void onNewIntent(Intent intent) {
1291        super.onNewIntent(intent);
1292        if(timer != null) {
1293            timer.cancel();
1294        }
1295    }
1296
1297    @Override
1298    protected void onLowMemory() {
1299        super.onLowMemory();
1300        if(timer != null) {
1301            timer.cancel();
1302        }
1303    }
1304
1305    @Override
1306    protected void onConfigurationChanged(Configuration newConfig) {
1307        super.onConfigurationChanged(newConfig);
1308        if(timer != null) {
1309            timer.start();
1310        }
1311    }
1312
1313    @Override
1314    protected void onTrimMemory(int level) {
1315        super.onTrimMemory(level);
1316        if(timer != null) {
1317            timer.cancel();
1318        }
1319    }
1320
1321    @Override
1322    protected void onNewIntent(Intent intent) {
1323        super.onNewIntent(intent);
1324        if(timer != null) {
1325            timer.cancel();
1326        }
1327    }
1328
1329    @Override
1330    protected void onLowMemory() {
1331        super.onLowMemory();
1332        if(timer != null) {
1333            timer.cancel();
1334        }
1335    }
1336
1337    @Override
1338    protected void onConfigurationChanged(Configuration newConfig) {
1339        super.onConfigurationChanged(newConfig);
1340        if(timer != null) {
1341            timer.start();
1342        }
1343    }
1344
1345    @Override
1346    protected void onTrimMemory(int level) {
1347        super.onTrimMemory(level);
1348        if(timer != null) {
1349            timer.cancel();
1350        }
1351    }
1352
1353    @Override
1354    protected void onNewIntent(Intent intent) {
1355        super.onNewIntent(intent);
1356        if(timer != null) {
1357            timer.cancel();
1358        }
1359    }
1360
1361    @Override
1362    protected void onLowMemory() {
1363        super.onLowMemory();
1364        if(timer != null) {
1365            timer.cancel();
1366        }
1367    }
1368
1369    @Override
1370    protected void onConfigurationChanged(Configuration newConfig) {
1371        super.onConfigurationChanged(newConfig);
1372        if(timer != null) {
1373            timer.start();
1374        }
1375    }
1376
1377    @Override
1378    protected void onTrimMemory(int level) {
1379        super.onTrimMemory(level);
1380        if(timer != null) {
1381            timer.cancel();
1382        }
1383    }
1384
1385    @Override
1386    protected void onNewIntent(Intent intent) {
1387        super.onNewIntent(intent);
1388        if(timer != null) {
1389            timer.cancel();
1390        }
1391    }
1392
1393    @Override
1394    protected void onLowMemory() {
1395        super.onLowMemory();
1396        if(timer != null) {
1397            timer.cancel();
1398        }
1399    }
1400
1401    @Override
1402    protected void onConfigurationChanged(Configuration newConfig) {
1403        super.onConfigurationChanged(newConfig);
1404        if(timer != null) {
1405            timer.start();
1406        }
1407    }
1408
1409    @Override
1410    protected void onTrimMemory(int level) {
1411        super.onTrimMemory(level);
1412        if(timer != null) {
1413            timer.cancel();
1414        }
1415    }
1416
1417    @Override
1418    protected void onNewIntent(Intent intent) {
1419        super.onNewIntent(intent);
1420        if(timer != null) {
1421            timer.cancel();
1422        }
1423    }
1424
1425    @Override
1426    protected void onLowMemory() {
1427        super.onLowMemory();
1428        if(timer != null) {
1429            timer.cancel();
1430        }
1431    }
1432
1433    @Override
1434    protected void onConfigurationChanged(Configuration newConfig) {
1435        super.onConfigurationChanged(newConfig);
1436        if(timer != null) {
1437            timer.start();
1438        }
1439    }
1440
1441    @Override
1442    protected void onTrimMemory(int level) {
1443        super.onTrimMemory(level);
1444        if(timer != null) {
1445            timer.cancel();
1446        }
1447    }
1448
1449    @Override
1450    protected void onNewIntent(Intent intent) {
1451        super.onNewIntent(intent);
1452        if(timer != null) {
1453            timer.cancel();
1454        }
1455    }
1456
1457    @Override
1458    protected void onLowMemory() {
1459        super.onLowMemory();
1460        if(timer != null) {
1461            timer.cancel();
1462        }
1463    }
1464
1465    @Override
1466    protected void onConfigurationChanged(Configuration newConfig) {
1467        super.onConfigurationChanged(newConfig);
1468        if(timer != null) {
1469            timer.start();
1470        }
1471    }
1472
1473    @Override
1474    protected void onTrimMemory(int level) {
1475        super.onTrimMemory(level);
1476        if(timer != null) {
1477            timer.cancel();
1478        }
1479    }
1480
1481    @Override
1482    protected void onNewIntent(Intent intent) {
1483        super.onNewIntent(intent);
1484        if(timer != null) {
1485            timer.cancel();
1486        }
1487    }
1488
1489    @Override
1490    protected void onLowMemory() {
1491        super.onLowMemory();
1492        if(timer != null) {
1493            timer.cancel();
1494        }
1495    }
1496
1497    @Override
1498    protected void onConfigurationChanged(Configuration newConfig) {
1499        super.onConfigurationChanged(newConfig);
1500        if(timer != null) {
1501            timer.start();
1502        }
1503    }
1504
1505    @Override
1506    protected void onTrimMemory(int level) {
1507        super.onTrimMemory(level);
1508        if(timer != null) {
1509            timer.cancel();
1510        }
1511    }
1512
1513    @Override
1514    protected void onNewIntent(Intent intent) {
1515        super.onNewIntent(intent);
1516        if(timer != null) {
1517            timer.cancel();
1518        }
1519    }
1520
1521    @Override
1522    protected void onLowMemory() {
1523        super.onLowMemory();
1524        if(timer != null) {
1525            timer.cancel();
1526        }
1527    }
1528
1529    @Override
1530    protected void onConfigurationChanged(Configuration newConfig) {
1531        super.onConfigurationChanged(newConfig);
1532        if(timer != null) {
1533            timer.start();
1534        }
1535    }
1536
1537    @Override
1538    protected void onTrimMemory(int level) {
1539        super.onTrimMemory(level);
1540        if(timer != null) {
1541            timer.cancel();
1542        }
1543    }
1544
1545    @Override
1546    protected void onNewIntent(Intent intent) {
1547        super.onNewIntent(intent);
1548        if(timer != null) {
1549            timer.cancel();
1550        }
1551    }
1552
1553    @Override
1554    protected void onLowMemory() {
1555        super.onLowMemory();
1556        if(timer != null) {
1557            timer.cancel();
1558        }
1559    }
1560
1561    @Override
1562    protected void onConfigurationChanged(Configuration newConfig) {
1563        super.onConfigurationChanged(newConfig);
1564        if(timer != null) {
1565            timer.start();
1566        }
1567    }
1568
1569    @Override
1570    protected void onTrimMemory(int level) {
1571        super.onTrimMemory(level);
1572        if(timer != null) {
1573            timer.cancel();
1574        }
1575    }
1576
1577    @Override
1578    protected void onNewIntent(Intent intent) {
1579        super.onNewIntent(intent);
1580        if(timer != null) {
1581            timer.cancel();
1582        }
1583    }
1584
1585    @Override
1586    protected void onLowMemory() {
1587        super.onLowMemory();
1588        if(timer != null) {
1589            timer.cancel();
1590        }
1591    }
1592
1593    @Override
1594    protected void onConfigurationChanged(Configuration newConfig) {
1595        super.onConfigurationChanged(newConfig);
1596        if(timer != null) {
1597            timer.start();
1598        }
1599    }
1600
1601    @Override
1602    protected void onTrimMemory(int level) {
1603        super.onTrimMemory(level
```

```
104             long seconds = totalSeconds % 60;
105
106             String time = String.format(Locale.getDefault(), format: "%02d:%02d:%02d", hours, minutes, seconds);
107             final String[] hourMinSec = time.split( regex: ":" );
108             TextView hour = findViewById(R.id.hour);
109             TextView min = findViewById(R.id.min);
110             TextView sec = findViewById(R.id.sec);
111             hour.setText(hourMinSec[0]);
112             min.setText(hourMinSec[1]);
113             sec.setText(hourMinSec[2]);
114         }
115
116         @Override
117         public void onFinish() {
118             startActivity(new Intent(getApplicationContext(), MainActivity.class));
119         }
120     }.start();
121
122     btnDone.setOnClickListener(view -> {
123         pb.setVisibility(LinearLayout.VISIBLE);
124         new Handler().postDelayed(new Runnable() {
125             @Override
126             public void run() {
127                 pb.setVisibility(LinearLayout.INVISIBLE);
128
129                 String urlReq = "http://192.168.100.4/ezgo/router.php";
130                 String method = "";
131                 String productID = "";
132                 String id = "";
133                 int updAmt = 0;
```

```
PaymentDetailActivity.java
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165

switch (type){
    case 1:
        method = "orderTicket";
        productID = tix.productID;
        id = tix.ticketID;
        updAmt = tix.tcSeat - amount;
        break;
    case 2:
        method = "orderHotel";
        productID = htl.productID;
        id = htl.hotelID;
        updAmt = htl.hNights - amount;
        break;
    case 3:
        method = "orderTour";
        productID = trp.productID;
        id = trp.tourID;
        updAmt = trp.tpSlot - amount;
        break;
}
}

SharedPreferences preferences = getSharedPreferences( name: "ezgo", Context.MODE_PRIVATE);
String userJson = preferences.getString( s: "user", s1: null);
User user = new Gson().fromJson(userJson, User.class);

RequestQueue queue = Volley.newRequestQueue( context: PaymentDetailActivity.this);
Gson gson = new Gson();

Map<String, Object> params = new HashMap<>();
params.put( k: "controller", v: "order");
params.put( k: "method", method);
```

```
PaymentDetailActivity.java
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196

params.put( k: "productID", productID);
params.put( k: "userID", user.userID);
params.put( k: "payMethod", bank);
params.put( k: "tdAmount", amount);
params.put( k: "tdTotalPrice", price);
params.put( k: "id", id);
params.put( k: "upAmount", updAmt);

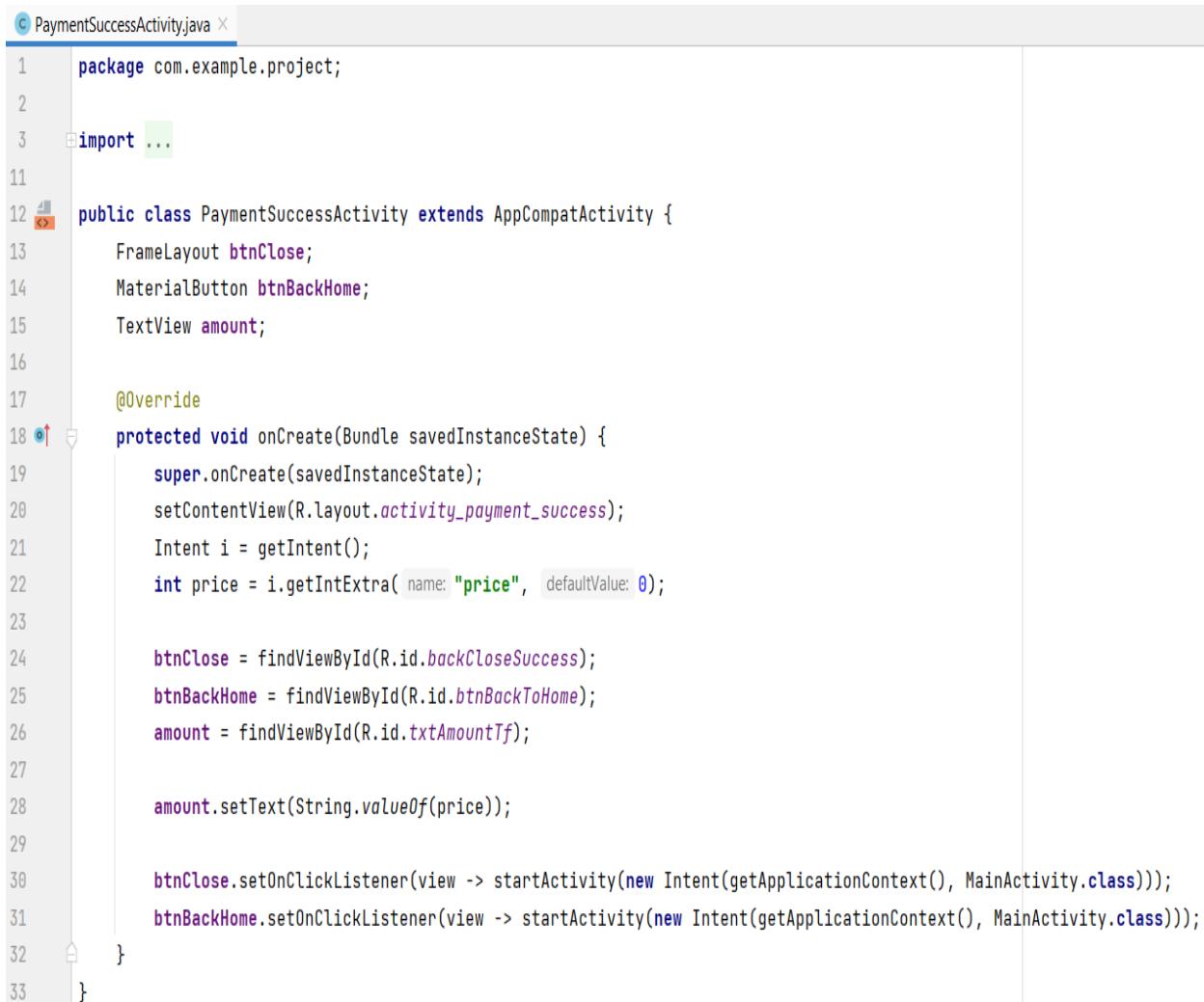
JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
    new JSONObject(params),
    new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject response) {
            Log.d( tag: "Ezgo", msg: "ResponseHome: " + response);
            ResponseNoObject resp = gson.fromJson(response.toString(), new TypeToken<ResponseNoObject>() {}.getType());
            boolean success = resp.isSuccess();

            if (success == true) {
                try {
                    Intent i = new Intent(getApplicationContext(), PaymentSuccessActivity.class);
                    i.putExtra( name: "price", price);
                    i.putExtra( name: "norek", noRek);
                    startActivity(i);
                    finish();
                } catch (Exception e){
                    Log.e( tag: "Ezgo", msg: "Error: " + e);
                }
            }
        }
    },
    new Response.ErrorListener() {
```

```
PaymentDetailActivity.java
197
198     @Override
199     public void onErrorResponse(VolleyError error) {
200         Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
201         String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
202         Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
203     });
204     queue.add(jsonObjectRequest);
205 }, delayMillis: 3000);
206 );
207 );
208
209 copyButtonRek.setOnClickListener(view -> {
210     ClipboardManager clipboardManager = (ClipboardManager) getSystemService(Context.CLIPBOARD_SERVICE);
211     ClipData clipData = ClipData.newPlainText( label: "No Rek", noRek);
212     clipboardManager.setPrimaryClip(clipData);
213     Toast.makeText(getApplicationContext(), text: "Teks disalin ke Clipboard", Toast.LENGTH_SHORT).show();
214 );
215
216 copyButtonAmount.setOnClickListener(view -> {
217     ClipboardManager clipboardManager = (ClipboardManager) getSystemService(Context.CLIPBOARD_SERVICE);
218     ClipData clipData = ClipData.newPlainText( label: "Price", String.valueOf(price));
219     clipboardManager.setPrimaryClip(clipData);
220     Toast.makeText(getApplicationContext(), text: "Teks disalin ke Clipboard", Toast.LENGTH_SHORT).show();
221 );
222 );
223 }
```

Kode java *PaymentDetailActivity* di atas merupakan implementasi dari *PaymentDetailActivity* dalam aplikasi Android yang digunakan untuk menampilkan detail pembayaran dan menyelesaikan transaksi. Aktivitas ini menampilkan informasi tentang nomor rekening bank, nama bank, jumlah yang harus dibayar, dan gambar bank. Informasi ini diterima dari *Intent* yang memulai aktivitas ini. Selain itu, aktivitas ini menampilkan hitungan mundur menggunakan *CountDownTimer* dan menyediakan tombol untuk menyalin informasi nomor rekening dan jumlah ke clipboard. Setelah pengguna menekan tombol “Done”, aktivitas ini mengirimkan permintaan POST ke server untuk menyelesaikan transaksi, memperbarui jumlah produk yang tersedia, dan mengarahkan pengguna ke *PaymentSuccessActivity* jika transaksi berhasil. Fitur penting lainnya adalah visibilitas *progress bar* saat memproses permintaan dan menangani respons dari server untuk memastikan transaksi selesai dengan benar.

### **3.2.34. PaymentSuccessActivity**

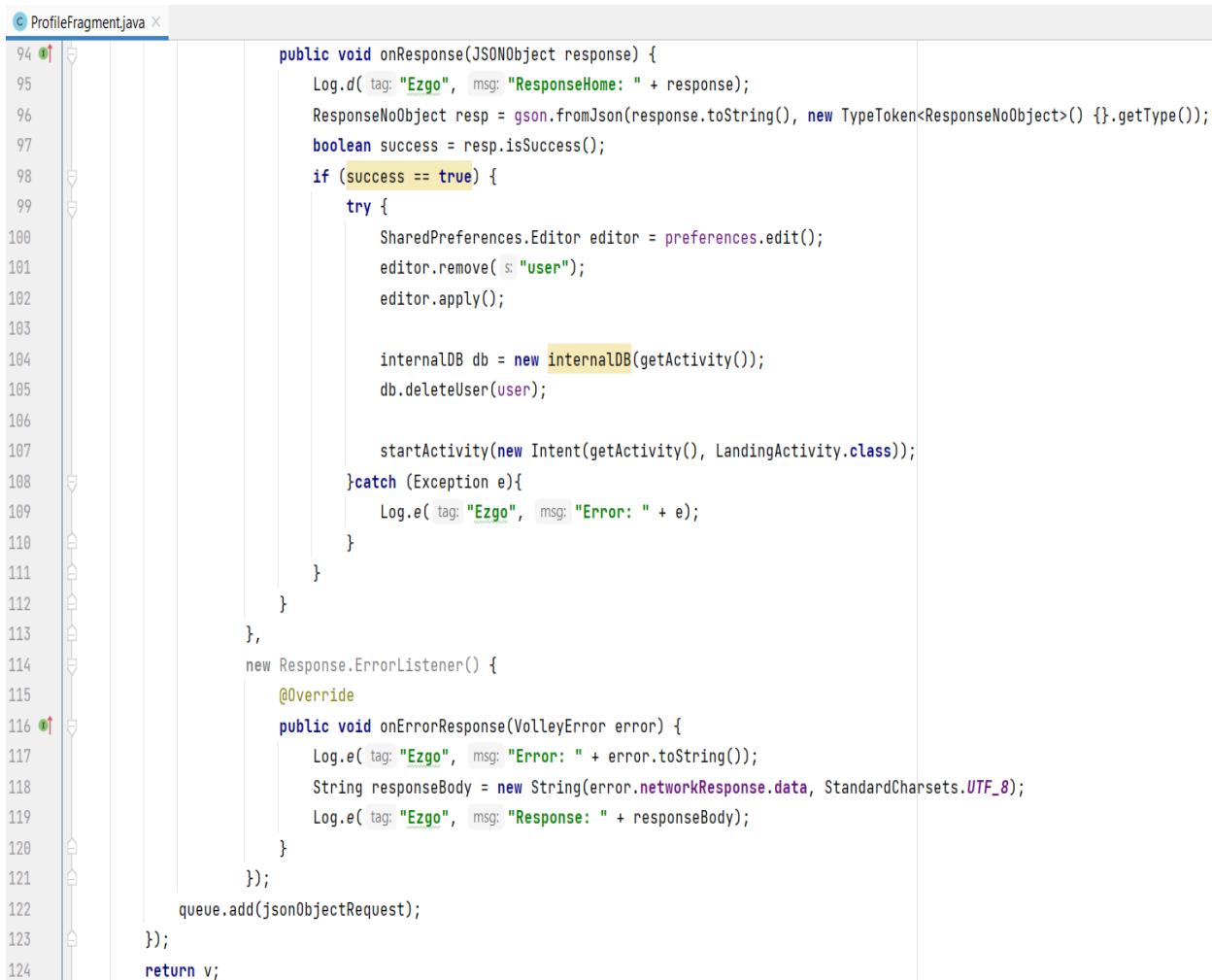


```
1 package com.example.project;
2
3 import ...
11
12 public class PaymentSuccessActivity extends AppCompatActivity {
13     FrameLayout btnClose;
14     MaterialButton btnBackHome;
15     TextView amount;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_payment_success);
21         Intent i = getIntent();
22         int price = i.getIntExtra("price", 0);
23
24         btnClose = findViewById(R.id.btnCloseSuccess);
25         btnBackHome = findViewById(R.id.btnCloseToHome);
26         amount = findViewById(R.id.txtAmountTf);
27
28         amount.setText(String.valueOf(price));
29
30         btnClose.setOnClickListener(view -> startActivity(new Intent(getApplicationContext(), MainActivity.class)));
31         btnBackHome.setOnClickListener(view -> startActivity(new Intent(getApplicationContext(), MainActivity.class)));
32     }
33 }
```

Kode java PaymentSuccessActivity di atas merupakan implementasi dari aktivitas Android yang menampilkan halaman konfirmasi pembayaran yang berhasil. Aktivitas ini menampilkan jumlah pembayaran yang berhasil dilakukan oleh pengguna. Jumlah tersebut diambil dari *Intent* yang memulai aktivitas ini. Terdapat dua tombol dalam halaman ini: *btnClose* dan *btnBackHome*. Keduanya memiliki fungsi yang sama yaitu mengarahkan pengguna kembali ke halaman utama (*MainActivity*). *btnClose* diaktifkan ketika pengguna menekan tombol tutup, sementara *btnBackHome* diaktifkan ketika pengguna menekan tombol kembali ke beranda. Dengan demikian, aktivitas ini memberikan konfirmasi visual kepada pengguna bahwa pembayaran telah berhasil dan memberikan navigasi kembali ke halaman utama aplikasi.

### 3.2.35. ProfileFragment

```
ProfileFragment.java
1 package com.example.project;
2
3 import ...
35
36 public class ProfileFragment extends Fragment {
37     TextView account, password, about, del, logout, name;
38     ShapeableImageView img;
39     String image;
40
41     @Nullable
42     @Override
43     public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
44         View v = inflater.inflate(R.layout.activity_profile, container, attachToRoot: false);
45         account = v.findViewById(R.id.btnAccount);
46         password = v.findViewById(R.id.btnPassword);
47         about = v.findViewById(R.id.btnAbout);
48         del = v.findViewById(R.id.btnDelAcc);
49         logout = v.findViewById(R.id.btnLogout);
50         name = v.findViewById(R.id.accName);
51         img = v.findViewById(R.id.accImg);
52
53         SharedPreferences preferences = getActivity().getSharedPreferences( name: "ezgo", Context.MODE_PRIVATE);
54         String userJson = preferences.getString( s: "user", s: null);
55         User user = new Gson().fromJson(userJson, User.class);
56
57         String urlReq = "http://192.168.100.4/ezgo/router.php";
58         String urlImg = "http://192.168.100.4/ezgo/images/";
59
60         name.setText(user.uName);
61         if(user.uProfilePicture != null){
62             image = urlImg + user.uProfilePicture;
63         }else{
64             image = urlImg + "defaultfp.png";
65         }
66         Picasso.get().load(image).into(img);
67
68         account.setOnClickListener(view -> startActivity(new Intent(getActivity(), AccountInformationActivity.class)));
69         password.setOnClickListener(view -> startActivity(new Intent(getActivity(), ChangePassActivity.class)));
70         about.setOnClickListener(view -> startActivity(new Intent(getActivity(), AboutActivity.class)));
71         logout.setOnClickListener(view -> {
72             SharedPreferences.Editor editor = preferences.edit();
73             editor.remove( s: "user");
74             editor.apply();
75
76             internalDB db = new internalDB(getActivity());
77             db.deleteUser(user);
78
79             startActivity(new Intent(getActivity(), LandingActivity.class));
80         });
81         del.setOnClickListener(view -> {
82             RequestQueue queue = Volley.newRequestQueue(requireActivity());
83             Gson gson = new Gson();
84
85             Map<String, Object> params = new HashMap<>();
86             params.put( k: "controller", v: "account");
87             params.put( k: "method", v: "delete");
88             params.put( k: "userID", user.userID);
89
90             JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
91                 new JSONObject(params),
92                 new Response.Listener<JSONObject>() {
93                     @Override
```



```
public void onResponse(JSONObject response) {
    Log.d( tag: "Ezgo", msg: "ResponseHome: " + response);
    ResponseNoObject resp = gson.fromJson(response.toString(), new TypeToken<ResponseNoObject>() {}.getType());
    boolean success = resp.isSuccess();
    if (success == true) {
        try {
            SharedPreferences.Editor editor = preferences.edit();
            editor.remove( s: "user");
            editor.apply();

            internalDB db = new internalDB(getActivity());
            db.deleteUser(user);

            startActivity(new Intent(getActivity(), LandingActivity.class));
        } catch (Exception e){
            Log.e( tag: "Ezgo", msg: "Error: " + e);
        }
    }
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
        String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
        Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
    }
);
queue.add(jsonObjectRequest);
});
return v;
```

Kode java ProfileFragment di atas merupakan implementasi dari sebuah fragment dalam aplikasi Android yang bertanggung jawab untuk menampilkan dan mengelola profil pengguna. *Fragment* ini memiliki beberapa fitur utama, yaitu melihat informasi akun, mengubah password, melihat informasi tentang aplikasi, menghapus akun, dan logout. Data pengguna diambil dari *SharedPreferences* dan digunakan untuk menampilkan nama dan gambar profil pengguna. Jika pengguna memilih untuk menghapus akun atau logout, data pengguna akan dihapus dari *SharedPreferences* dan database internal aplikasi (*internalDB*), kemudian pengguna akan diarahkan kembali ke halaman landing (*LandingActivity*). Selain itu, fragment ini juga menyediakan navigasi ke aktivitas lain untuk memperbarui informasi akun, mengubah password, dan melihat informasi tentang aplikasi. Implementasi ini memastikan bahwa pengguna dapat dengan mudah mengelola informasi profil mereka serta keluar dari aplikasi atau menghapus akun mereka jika diperlukan.

### 3.2.36. ResponseFourObjectList

```
c ResponseFourObjectList.java ×
1  package com.example.project;
2
3  import java.util.List;
4
5  public class ResponseFourObjectList<T, U, V, W> {
6      private boolean Success;
7      private List<T> Data1;
8      private List<U> Data2;
9      private List<V> Data3;
10     private List<W> Data4;
11
12     public boolean isSuccess() { return Success; }
13
14     public List<T> getData1() { return Data1; }
15
16     public List<U> getData2() { return Data2; }
17
18     public List<V> getData3() { return Data3; }
19
20     public List<W> getData4() { return Data4; }
21
22 }
```

Kode java ResponseFourObjectList di atas merupakan sebuah kelas generik dalam bahasa Java yang digunakan untuk menangani respons yang mengandung empat jenis objek berbeda dari suatu *API*. Kelas ini memiliki atribut untuk menandakan keberhasilan (*Success*) dan empat daftar objek (*Data1*, *Data2*, *Data3*, *Data4*) yang masing-masing dapat berisi tipe data yang berbeda. Metode *getter* disediakan untuk mengakses nilai dari atribut-atribut tersebut. Kelas ini berguna dalam situasi di mana respons API mengembalikan beberapa jenis data yang berbeda dalam satu respons, memungkinkan pengembang untuk dengan mudah mengelola dan memproses data yang diterima dari server dalam aplikasi.

### 3.2.37. ResponseMessage

```
c ResponseMessage.java ×
1  package com.example.project;
2
3  public class ResponseMessage {
4      private boolean Success;
5      private int Message;
6
7      public boolean isSuccess() { return Success; }
8
9      public int getMessage() { return Message; }
10
11 }
12
13 }
```

Kode java ResponseMessage di atas merupakan sebuah kelas Java yang digunakan untuk menangani respons sederhana dari suatu *API*. Kelas ini memiliki dua atribut utama: *Success*, yang merupakan boolean untuk menandakan apakah operasi berhasil atau tidak, dan *Message*, yang merupakan integer untuk menyimpan pesan atau kode status tambahan terkait respons tersebut. Kelas ini menyediakan metode *getter* untuk mengakses nilai dari kedua atribut tersebut. Fungsi utama dari kelas ini adalah untuk menyederhanakan pengelolaan respons dari server, memungkinkan aplikasi untuk dengan mudah menentukan hasil dari suatu operasi dan mengambil tindakan yang sesuai berdasarkan status dan pesan yang diterima.

### 3.2.38. ResponseNoObject

```
c ResponseNoObject.java ×
1  package com.example.project;
2
3  public class ResponseNoObject {
4      private boolean Success;
5
6      public boolean isSuccess() { return Success; }
7
8  }
```

Kode java ResponseNoObject di atas merupakan sebuah kelas Java yang digunakan untuk menangani respons sederhana dari suatu *API*. Kelas ini hanya memiliki satu atribut *boolean* bernama *Success*, yang digunakan untuk menandakan apakah operasi yang dilakukan berhasil atau tidak. Kelas ini menyediakan metode *getter isSuccess()* untuk mengakses nilai dari atribut *Success*. Fungsi utama dari kelas ini adalah untuk menyederhanakan pengelolaan respons dari server yang tidak memerlukan data tambahan selain status keberhasilan operasi. Dengan menggunakan kelas ini, aplikasi dapat dengan mudah menentukan hasil dari suatu operasi dan mengambil tindakan yang sesuai berdasarkan status sukses yang diterima.

### 3.2.39. ResponseOneObject

```
c ResponseOneObject.java ×
1 package com.example.project;
2
3 public class ResponseOneObject<T> {
4     private boolean Success;
5     private T Data;
6
7     public boolean isSuccess() { return Success; }
8
9     public T getData() { return Data; }
10
11 }
12
13 }
```

Kode java ResponseOneObject di atas merupakan sebuah kelas Java generik yang digunakan untuk menangani respons dari *API* yang mengembalikan satu objek data. Kelas ini memiliki dua atribut: *Success* yang merupakan boolean untuk menandakan apakah operasi berhasil atau tidak, dan *Data* yang merupakan objek dari tipe *generik T* yang menampung data yang dikembalikan dari server. Kelas ini menyediakan dua metode *getter*, *isSuccess()* untuk mengakses nilai dari *Success* dan *getData()* untuk mengakses nilai dari *Data*. Fungsi utama dari kelas ini adalah untuk mempermudah pengelolaan respons dari server yang mengembalikan satu objek data, sehingga aplikasi dapat dengan mudah mengambil dan memproses data yang diterima serta menentukan status operasi berdasarkan nilai sukses yang diterima.

### 3.2.40. ResponseOneObjectList

```
c ResponseOneObjectList.java ×
1 package com.example.project;
2
3 import java.util.List;
4
5 public class ResponseOneObjectList<T> {
6     private boolean Success;
7     private List<T> Data;
8
9     public boolean isSuccess() { return Success; }
10
11     public List<T> getData() { return Data; }
12
13 }
14
15 }
```

Kode java ResponseOneObjectList di atas merupakan sebuah kelas Java generik yang digunakan untuk menangani respons dari *API* yang mengembalikan sebuah daftar (list) objek

data. Kelas ini memiliki dua atribut utama: *Success* yang merupakan boolean untuk menandakan apakah operasi berhasil atau tidak, dan *Data* yang merupakan daftar objek dari tipe *generik T* yang menampung data yang dikembalikan dari server. Kelas ini menyediakan dua metode *getter*, *isSuccess()* untuk mengakses nilai dari *Success* dan *getData()* untuk mengakses nilai dari *Data*. Fungsi utama dari kelas ini adalah untuk mempermudah pengelolaan respons dari server yang mengembalikan daftar objek data, sehingga aplikasi dapat dengan mudah mengambil dan memproses data yang diterima serta menentukan status operasi berdasarkan nilai sukses yang diterima.

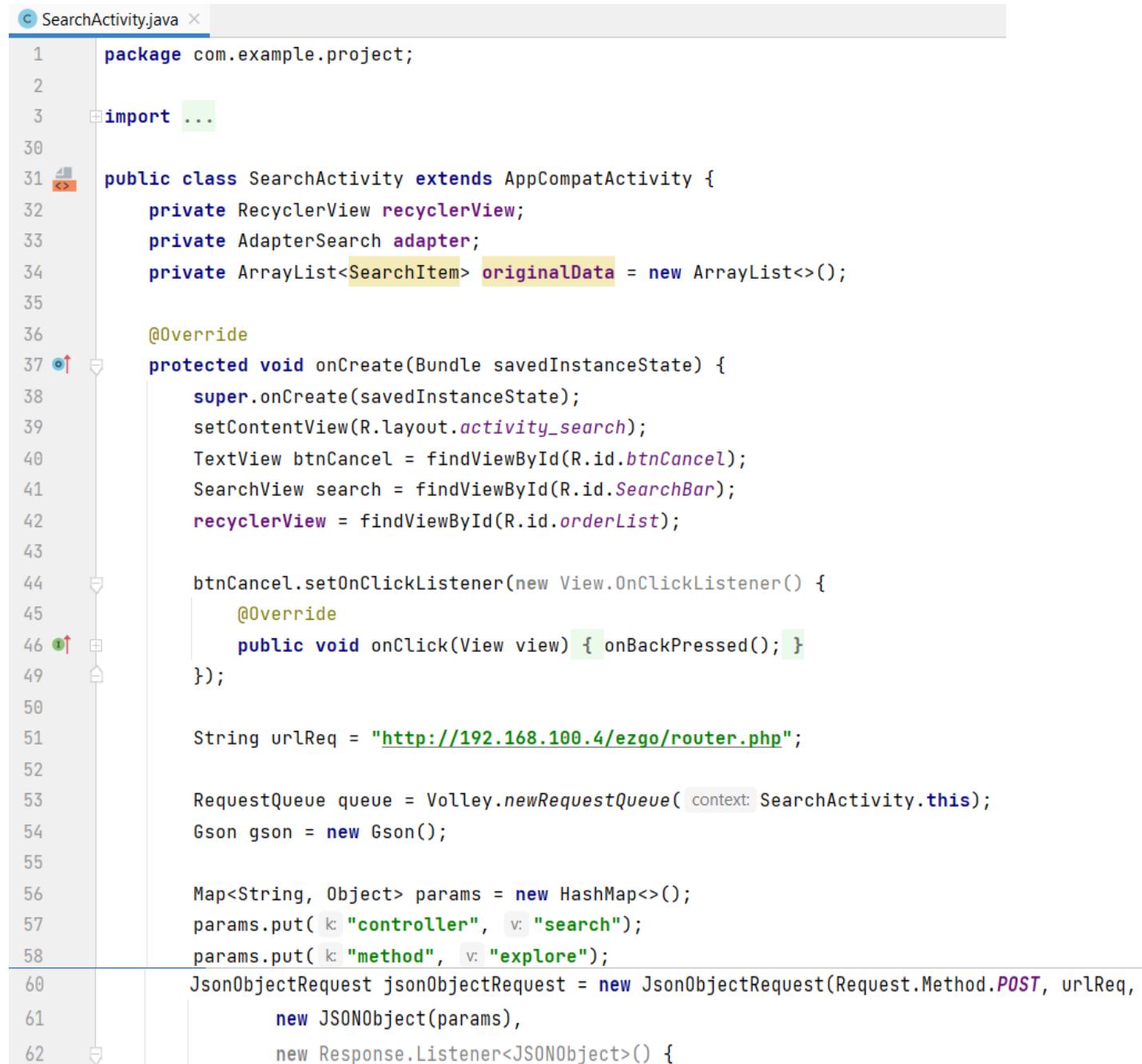
### 3.2.41. ResponseThreeObjectList

```
 1  package com.example.project;
 2
 3  import java.util.List;
 4
 5  public class ResponseThreeObjectList<T, U, V>{
 6      private boolean Success;
 7      private List<T> Data1;
 8      private List<U> Data2;
 9      private List<V> Data3;
10
11      public boolean isSuccess() { return Success; }
12
13
14      public List<T> getData1() { return Data1; }
15
16      public List<U> getData2() { return Data2; }
17
18      public List<V> getData3() { return Data3; }
19
20  }
```

Kode java ResponseThreeObjectList di atas merupakan sebuah kelas Java generik yang digunakan untuk menangani respons dari *API* yang mengembalikan tiga daftar objek data. Kelas ini memiliki beberapa atribut utama: *Success* yang merupakan boolean untuk menandakan apakah operasi berhasil atau tidak, *Data1* yang merupakan daftar objek dari tipe *generik T*, *Data2* yang merupakan daftar objek dari tipe *generik U*, dan *Data3* yang merupakan daftar objek dari tipe *generik V*. Kelas ini menyediakan metode *getter* untuk masing-masing atribut, yaitu *isSuccess()* untuk mengakses nilai dari *Success*, *getData1()* untuk mengakses nilai dari *Data1*, *getData2()* untuk mengakses nilai dari *Data2*, dan *getData3()* untuk mengakses nilai dari *Data3*. Fungsi utama dari kelas ini adalah untuk mempermudah pengelolaan respons dari server yang

mengembalikan beberapa daftar objek data, sehingga aplikasi dapat dengan mudah mengambil dan memproses data yang diterima serta menentukan status operasi berdasarkan nilai sukses yang diterima.

### 3.2.42. SearchActivity



```
1 package com.example.project;
2
3 import ...
30
31 public class SearchActivity extends AppCompatActivity {
32     private RecyclerView recyclerView;
33     private AdapterSearch adapter;
34     private ArrayList<SearchItem> originalData = new ArrayList<>();
35
36     @Override
37     protected void onCreate(Bundle savedInstanceState) {
38         super.onCreate(savedInstanceState);
39         setContentView(R.layout.activity_search);
40         TextView btnCancel = findViewById(R.id.btnCancel);
41         SearchView search = findViewById(R.id.SearchBar);
42         recyclerView = findViewById(R.id.orderList);
43
44         btnCancel.setOnClickListener(new View.OnClickListener() {
45             @Override
46             public void onClick(View view) { onBackPressed(); }
47         });
48
49         String urlReq = "http://192.168.100.4/ezgo/router.php";
50
51         RequestQueue queue = Volley.newRequestQueue( context: SearchActivity.this );
52         Gson gson = new Gson();
53
54         Map<String, Object> params = new HashMap<>();
55         params.put( k: "controller", v: "search" );
56         params.put( k: "method", v: "explore" );
57         JsonObjectRequest json0bjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
58             new JSONObject(params),
59             new Response.Listener<JSONObject>() {
```

```
 63  @Override
64  public void onResponse(JSONObject response) {
65      Log.d( tag: "Ezgo" , msg: "ResponseHome: " + response);
66      ResponseFourObjectList<location, ticket, hotel, tour> resp = gson.fromJson(response.toString(), new TypeToken<ResponseFourObjectList<location, ticket, hotel, tour>>() {}.get());
67      boolean success = resp.isSuccess();
68      List<location> locs = resp.getData1();
69      List<ticket> tixs = resp.getData2();
70      List<hotel> htls = resp.getData3();
71      List<tour> trps = resp.getData4();
72
73      if (success == true) {
74          try {
75              originalData.clear();
76              int num = 1;
77              for (location loc : locs) {
78                  originalData.add(new SearchItem(loc));
79                  Log.d( tag: "Ezgo" , msg: "currently: " + num);
80                  num++;
81              }
82
83              for (ticket tix : tixs) {
84                  originalData.add(new SearchItem(tix));
85                  Log.d( tag: "Ezgo" , msg: "currently: " + num);
86                  num++;
87              }
88
89              for (hotel htl : htls) {
90                  originalData.add(new SearchItem(htl));
91                  Log.d( tag: "Ezgo" , msg: "currently: " + num);
92                  num++;
93              }
94          }
95
96          for (tour trp : trps) {
97              originalData.add(new SearchItem(trp));
98              Log.d( tag: "Ezgo" , msg: "currently: " + num);
99              num++;
100         }
101
102         adapter = new AdapterSearch(originalData, context: SearchActivity.this);
103         recyclerView.setLayoutManager(new GridLayoutManager( context: SearchActivity.this, spanCount: 2));
104         recyclerView.setAdapter(adapter);
105     }catch (Exception e){
106         Log.e( tag: "Ezgo" , msg: "Error: " + e);
107     }
108 }
109
110 new Response.ErrorListener() {
111     @Override
112     public void onErrorResponse(VolleyError error) {
113         Log.e( tag: "Ezgo" , msg: "Error: " + error.toString());
114         String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
115         Log.e( tag: "Ezgo" , msg: "Response: " + responseBody);
116     }
117 });
118 queue.add(jsonObjectRequest);
119
120 search.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
121     @Override
122     public boolean onQueryTextSubmit(String query) { return false; }
123
124     @Override
125     public boolean onQueryTextChange(String newText) {
126
127 }
```

```
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
```

```

  SearchActivity.java ×
128             filterData(newText);
129             return true;
130         }
131     });
132 }
133
134     private void filterData(String query) {
135         // Filter data based on the user's input and update the adapter
136         ArrayList<SearchItem> filteredData = new ArrayList<>();
137         for (SearchItem item : originalData) {
138             if (item.getTitle().toLowerCase().contains(query.toLowerCase())) {
139                 filteredData.add(item);
140             }
141         }
142         adapter.setData(filteredData);
143     }
144 }

```

Kode java SearchActivity di atas merupakan sebuah kelas Java untuk aktivitas pencarian dalam aplikasi Android. Aktivitas ini memungkinkan pengguna untuk mencari berbagai item seperti lokasi, tiket, hotel, dan paket tour menggunakan *SearchView*. Ketika aktivitas ini dibuat, permintaan JSON dikirim ke server untuk mengambil data yang relevan, yang kemudian diubah menjadi objek dan ditambahkan ke *originalData*. Data ini kemudian ditampilkan dalam *RecyclerView* menggunakan *AdapterSearch*. Fitur pencarian diimplementasikan melalui *SearchView.OnQueryTextListener* yang memfilter data berdasarkan input pengguna dan memperbarui tampilan *RecyclerView* sesuai dengan hasil pencarian. Kelas ini juga menangani kesalahan jaringan dan menampilkan data hasil pencarian dalam grid layout dua kolom.

### 3.2.43. SearchItem

```

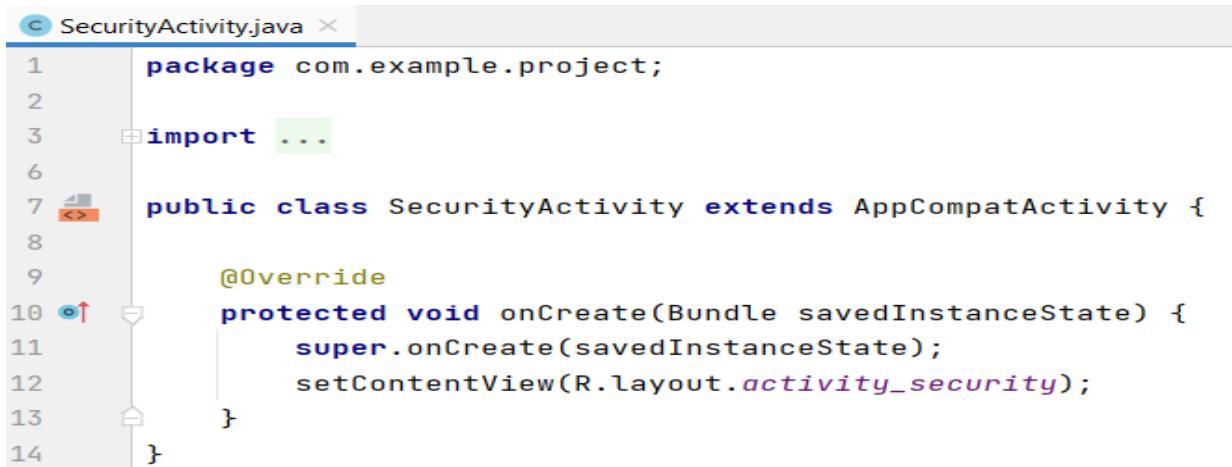
  SearchItem.java ×
1 package com.example.project;
2
3 public class SearchItem<T> {
4     private T obj;
5
6     public SearchItem(T obj) { this.obj = obj; }
7
8     public String getTitle() {
9         if (obj instanceof location) {
10             return ((location) obj).lName;
11         }
12     }

```

```
  SearchItem.java x
13     }else if(obj instanceof ticket){
14         return ((ticket) obj).tcName;
15     }else if(obj instanceof hotel){
16         return ((hotel) obj).hName;
17     }else if(obj instanceof tour){
18         return ((tour) obj).tpName;
19     }else{
20         return null;
21     }
22 }
23
24     public String getType() {
25         if (obj instanceof location) {
26             return "Destination";
27         }else if(obj instanceof ticket){
28             return "Ticket";
29         }else if(obj instanceof hotel){
30             return "Hotel";
31         }else if(obj instanceof tour){
32             return "Tour Package";
33         }else{
34             return null;
35         }
36     }
37
38     public Class<?> getObjectType() { return obj.getClass(); }
39
40     public T getObj() { return obj; }
41
42 }
```

Kode java SearchItem di atas merupakan sebuah kelas Java untuk mengelola objek pencarian dalam aplikasi Android. Kelas ini menggunakan *generik (T)* untuk menangani berbagai jenis objek, seperti lokasi (*location*), tiket (*ticket*), hotel (*hotel*), dan paket tour (*tour*). Fungsi utamanya adalah menyediakan metode untuk mendapatkan judul (*getTitle*), jenis objek (*getType*), dan tipe kelas objek (*getObjectType*). Hal ini memungkinkan data yang berbeda ditangani secara seragam dalam fitur pencarian dan ditampilkan dengan benar dalam antarmuka pengguna. Implementasi tersebut penting untuk memfasilitasi pencarian dan pengelompokan berbagai jenis data dalam aplikasi.

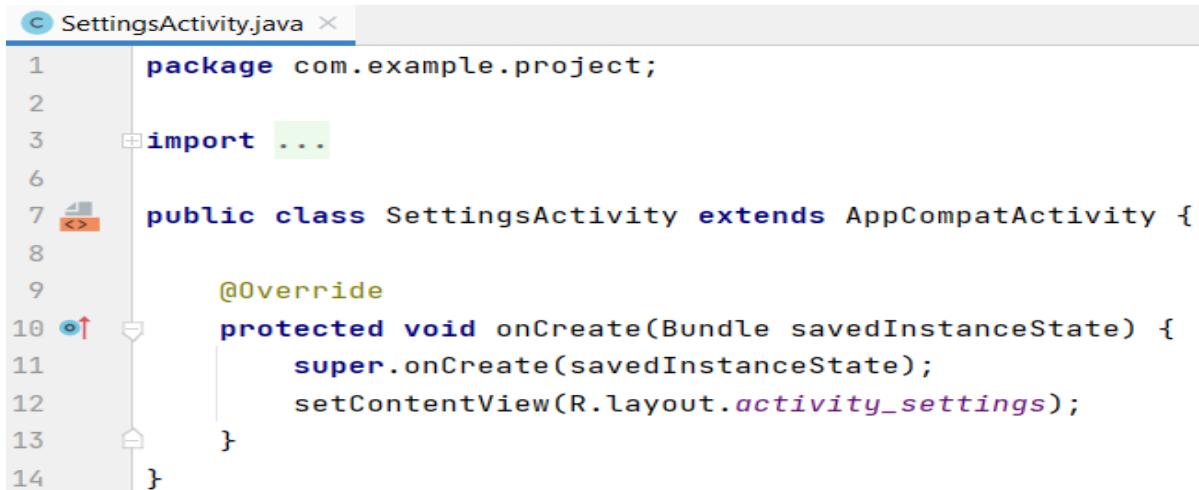
### 3.2.44. SecurityActivity



```
1 package com.example.project;
2
3 import ...
4
5
6
7 public class SecurityActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_security);
13    }
14}
```

Kode java SecurityActivity di atas merupakan sebuah kelas dalam aplikasi Android yang mewakili aktivitas keamanan. Kelas ini memperluas *AppCompatActivity* dan berfungsi untuk mengatur tampilan dan perilaku halaman keamanan aplikasi. Pada metode *onCreate*, layout *activity\_security* diatur sebagai konten tampilan aktivitas ini menggunakan *setContentView*. Fungsinya hanya untuk menampilkan layout keamanan yang telah didefinisikan di file XML terkait (*activity\_security.xml*).

### 3.2.45. SettingsActivity

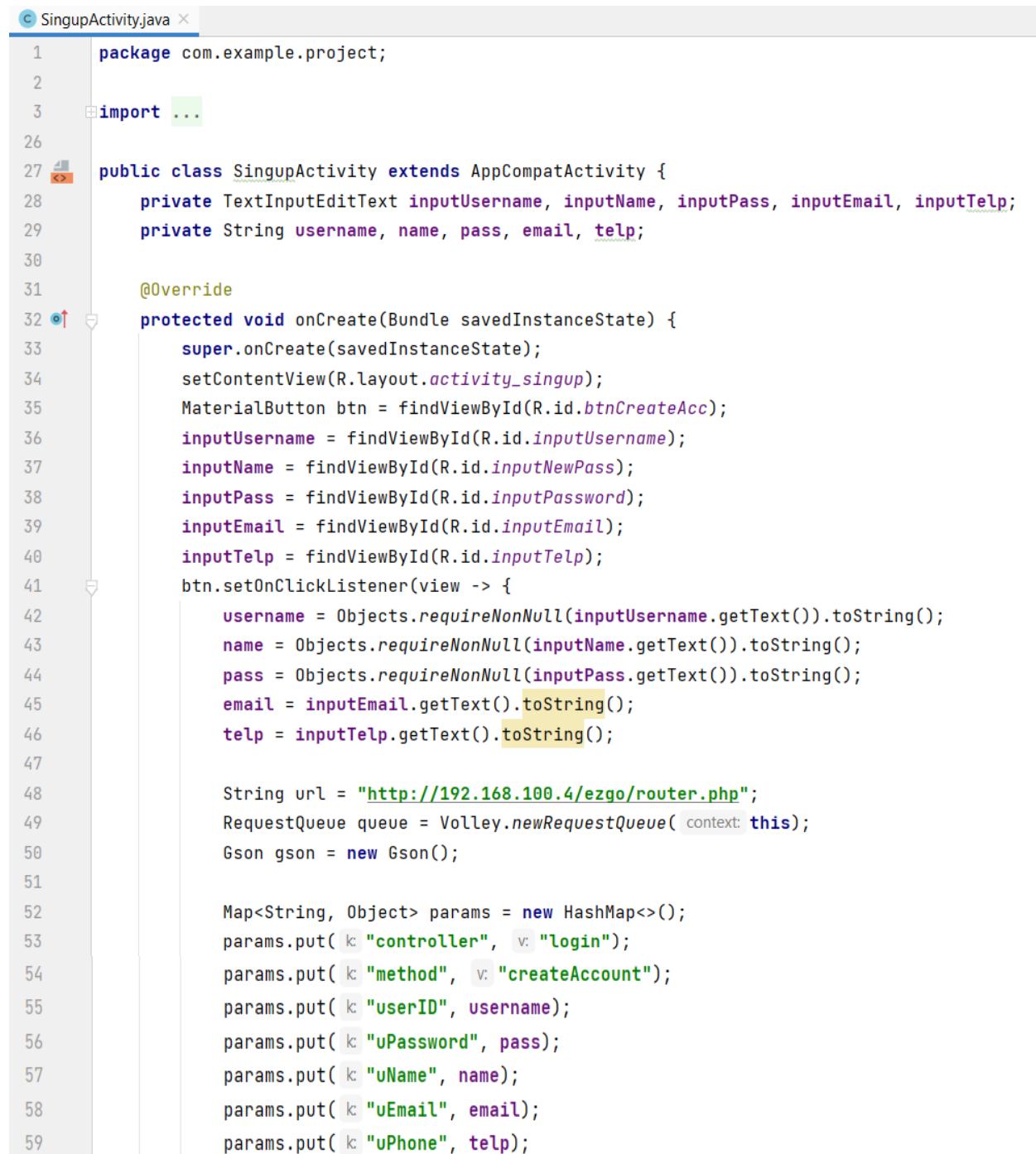


```
1 package com.example.project;
2
3 import ...
4
5
6
7 public class SettingsActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_settings);
13    }
14}
```

Kode java SettingsActivity di atas merupakan sebuah kelas dalam aplikasi Android yang mewakili aktivitas pengaturan. Kelas ini memperluas *AppCompatActivity* dan berfungsi untuk mengatur tampilan dan perilaku halaman pengaturan aplikasi. Pada metode *onCreate*, layout *activity\_settings* diatur sebagai konten tampilan aktivitas ini menggunakan *setContentView*.

Fungsinya hanya untuk menampilkan layout pengaturan yang telah didefinisikan di file XML terkait (*activity\_settings.xml*).

### 3.2.46. SingupActivity



```
1 package com.example.project;
2
3 import ...
4
5
6 public class SingupActivity extends AppCompatActivity {
7     private TextInputEditText inputUsername, inputName, inputPass, inputEmail, inputTelp;
8     private String username, name, pass, email, telp;
9
10
11    @Override
12    protected void onCreate(Bundle savedInstanceState) {
13        super.onCreate(savedInstanceState);
14        setContentView(R.layout.activity_singup);
15        MaterialButton btn = findViewById(R.id.btnCreateAcc);
16        inputUsername = findViewById(R.id.inputUsername);
17        inputName = findViewById(R.id.inputNewPass);
18        inputPass = findViewById(R.id.inputPassword);
19        inputEmail = findViewById(R.id.inputEmail);
20        inputTelp = findViewById(R.id.inputTelp);
21        btn.setOnClickListener(view -> {
22            username = Objects.requireNonNull(inputUsername.getText()).toString();
23            name = Objects.requireNonNull(inputName.getText()).toString();
24            pass = Objects.requireNonNull(inputPass.getText()).toString();
25            email = inputEmail.getText().toString();
26            telp = inputTelp.getText().toString();
27
28            String url = "http://192.168.100.4/ezgo/router.php";
29            RequestQueue queue = Volley.newRequestQueue( context: this );
30            Gson gson = new Gson();
31
32            Map<String, Object> params = new HashMap<>();
33            params.put( k: "controller", v: "login" );
34            params.put( k: "method", v: "createAccount" );
35            params.put( k: "userID", username );
36            params.put( k: "uPassword", pass );
37            params.put( k: "uName", name );
38            params.put( k: "uEmail", email );
39            params.put( k: "uPhone", telp );
40
41        });
42    }
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
```

```
© SingupActivity.java x
61
62
63
64
65 ①↑
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91

JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, url,
        new JSONObject(params),
        new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                ResponseMessage resp = gson.fromJson(response.toString(), ResponseMessage.class);
                boolean success = resp.isSuccess();
                Log.d( tag: "Ezgo", msg: "Response: " + success);

                if (success) {
                    /*if (errLogin != null) {
                        errLogin.setVisibility(View.GONE);
                   }*/
                    try {
                        User user = new User(username, pass, name, email, telp);
                        Log.d( tag: "Ezgo", msg: "user: " + user.userID);

                        internalDB db = new internalDB( context: SingupActivity.this);
                        db.createUser(user);

                        Log.d( tag: "Ezgo", msg: "SQLite created");

                        SharedPreferences preferences = getSharedPreferences( name: "ezgo", MODE_PRIVATE);
                        SharedPreferences.Editor editor = preferences.edit();
                        editor.putString( s: "user", user.toJson());
                        editor.apply();

                        Log.d( tag: "Ezgo", msg: "Preferences created");
                    }catch (Exception e){
                        Log.e( tag: "Ezgo", msg: "Error: " + e);
                    }
                }
            }
        }

© SingupActivity.java x
93
94
95
96
97
98
99
100
101
102
103
104 ①↑
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120

        Intent i = new Intent(getApplicationContext(), MainActivity.class);
        startActivity(i);
    } else {
        /*if (errLogin != null) {
            errLogin.setVisibility(View.VISIBLE);
       }*/
    }
},
new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
        String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
        Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
    }
};

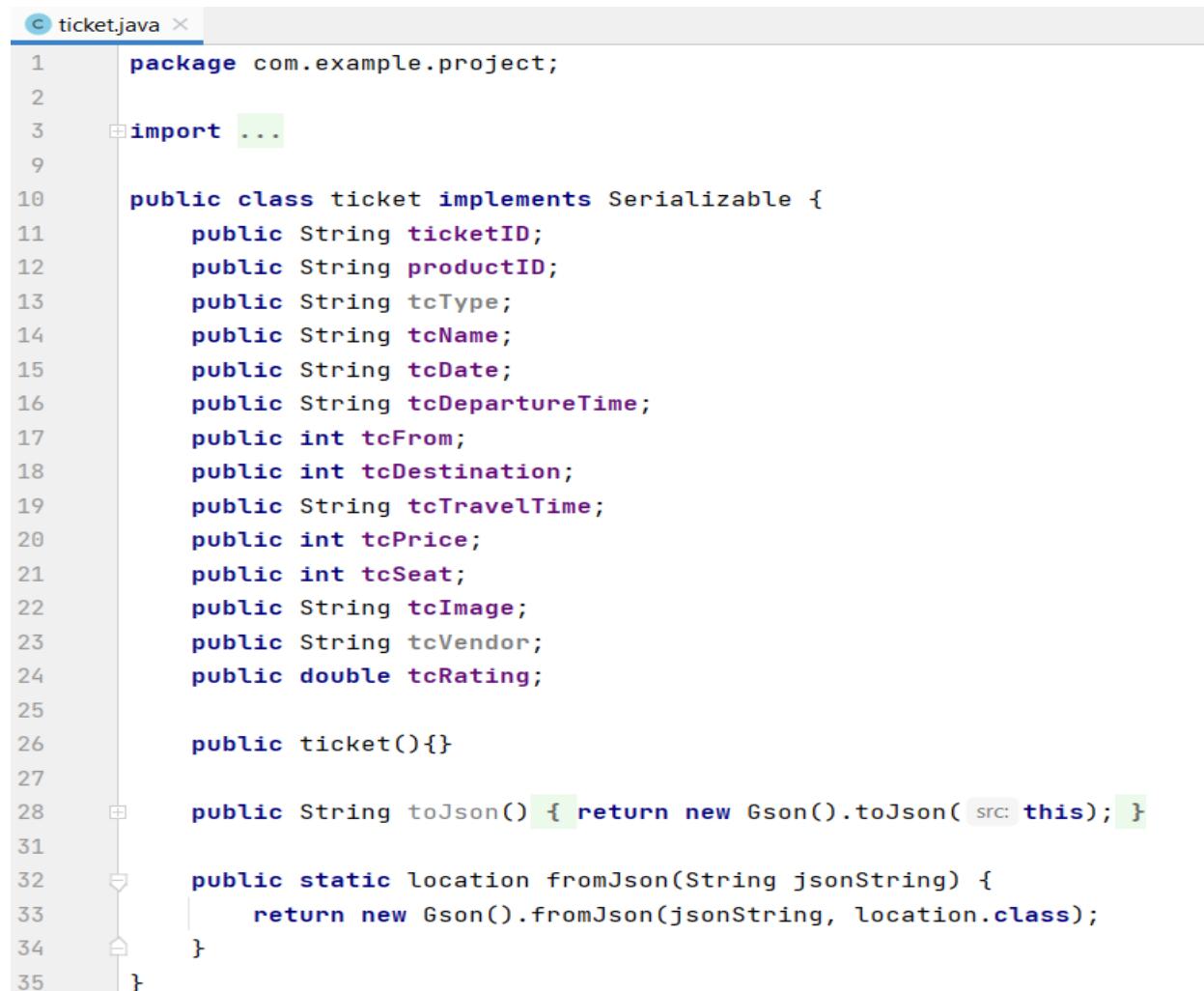
queue.add(jsonObjectRequest);

inputUsername.setText("");
inputName.setText("");
inputPass.setText("");
inputEmail.setText("");
inputTelp.setText("");
};

}
}
```

Kode java SignupActivity di atas merupakan sebuah kelas dalam aplikasi Android yang mewakili aktivitas pendaftaran pengguna baru. Kelas ini memperluas *AppCompatActivity* dan berfungsi untuk mengelola proses pendaftaran akun baru. Pada metode *onCreate*, layout *activity\_signup* diatur sebagai konten tampilan aktivitas ini menggunakan *setContentView*. Dalam kelas ini, beberapa input pengguna ditangani, seperti *username*, *name*, *password*, *email*, dan *telp*, yang kemudian dikirimkan ke server menggunakan *Volley* untuk membuat akun baru. Jika pendaftaran berhasil, data pengguna disimpan di database internal dan *SharedPreferences*, lalu pengguna diarahkan ke *MainActivity*. Kode ini juga mencakup penanganan error untuk memastikan bahwa respon dari server ditangani dengan baik.

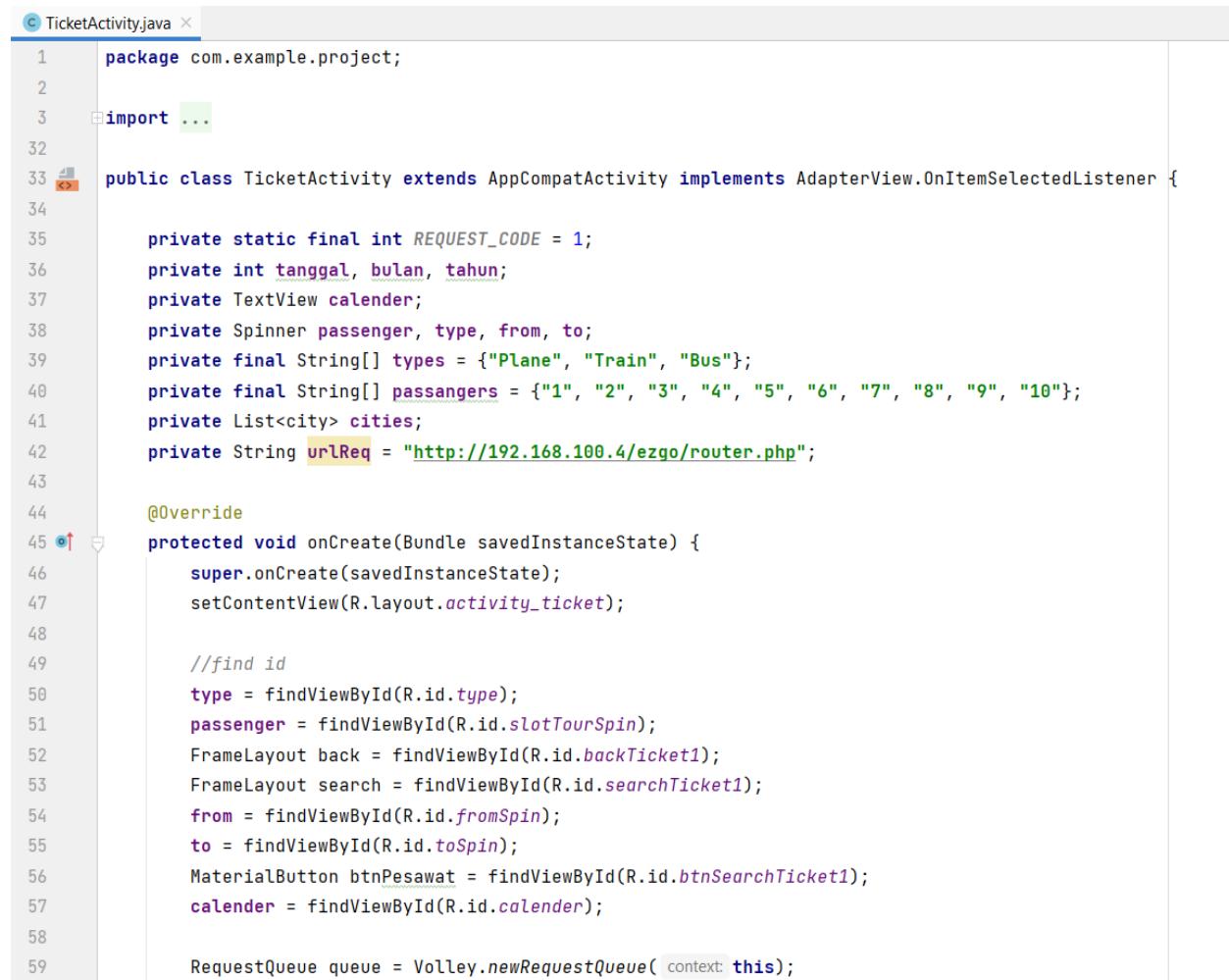
### **3.2.47. ticket**



```
ticket.java
1 package com.example.project;
2
3 import ...
4
5
6 public class ticket implements Serializable {
7     public String ticketID;
8     public String productID;
9     public String tcType;
10    public String tcName;
11    public String tcDate;
12    public String tcDepartureTime;
13    public int tcFrom;
14    public int tcDestination;
15    public String tcTravelTime;
16    public int tcPrice;
17    public int tcSeat;
18    public String tcImage;
19    public String tcVendor;
20    public double tcRating;
21
22    public ticket(){}
23
24    public String toJson() { return new Gson().toJson( this ); }
25
26    public static ticket fromJson(String jsonString) {
27        return new Gson().fromJson(jsonString, ticket.class);
28    }
29
30
31 }
```

Kode java ticket di atas merupakan sebuah kelas model dalam aplikasi Android yang digunakan untuk merepresentasikan data tiket. Kelas ini mengimplementasikan antarmuka *Serializable* agar objek *ticket* dapat dengan mudah dikirimkan antar aktivitas atau disimpan dalam bentuk serialized. Kelas ini memiliki beberapa atribut seperti *ticketID*, *productID*, *tcType*, *tcName*, *tcDate*, *tcDepartureTime*, *tcFrom*, *tcDestination*, *tcTravelTime*, *tcPrice*, *tcSeat*, *tcImage*, *tcVendor*, dan *tcRating* yang menyimpan informasi detail tentang tiket. Kelas ini juga menyediakan dua metode: *toJson()* untuk mengonversi objek ticket menjadi format JSON dan *fromJson(String jsonString)* untuk mengonversi string JSON kembali menjadi objek *location*. Kelas ini penting dalam pengelolaan data tiket di aplikasi, memudahkan penyimpanan, pengambilan, dan transfer data.

### 3.2.48. TicketActivity



```
1 package com.example.project;
2
3 import ...
32
33 public class TicketActivity extends AppCompatActivity implements AdapterView.OnItemClickListener {
34
35     private static final int REQUEST_CODE = 1;
36     private int tanggal, bulan, tahun;
37     private TextView calender;
38     private Spinner passenger, type, from, to;
39     private final String[] types = {"Plane", "Train", "Bus"};
40     private final String[] passangers = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10"};
41     private List<city> cities;
42     private String urlReq = "http://192.168.100.4/ezgo/router.php";
43
44     @Override
45     protected void onCreate(Bundle savedInstanceState) {
46         super.onCreate(savedInstanceState);
47         setContentView(R.layout.activity_ticket);
48
49         //find id
50         type = findViewById(R.id.type);
51         passenger = findViewById(R.id.slotTourSpin);
52         FrameLayout back = findViewById(R.id.backTicket1);
53         FrameLayout search = findViewById(R.id.searchTicket1);
54         from = findViewById(R.id.fromSpin);
55         to = findViewById(R.id.toSpin);
56         MaterialButton btnPesawat = findViewById(R.id.btnSearchTicket1);
57         calender = findViewById(R.id.calendar);
58
59         RequestQueue queue = Volley.newRequestQueue( context: this);
```

```
60     Gson gson = new Gson();
61
62     Map<String, Object> params = new HashMap<>();
63     params.put("controller", "city");
64     params.put("method", "list");
65
66     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
67         new JSONObject(params),
68         response -> {
69             Log.d("Ezgo", "ResponseHome: " + response);
70             ResponseOneObjectList<city> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObjectList<city>>() {
71                 .getType());
72             boolean success = resp.isSuccess();
73             cities = resp.getData();
74
75             if (success == true) {
76                 try {
77                     List<String> cityNames = new ArrayList<>();
78
79                     for (city city : cities) {
80                         cityNames.add(city.cName);
81                         Log.d("Ezgo", "ResponseHome: " + cityNames);
82                     }
83
84                     ArrayAdapter<String> adapterFrom = new ArrayAdapter<>(context: TicketActivity.this, android.R.layout.simple_spinner_item, cityNames);
85                     from.setAdapter(adapterFrom);
86                     from.setOnItemSelectedListener(TicketActivity.this);
87
88                     ArrayAdapter<String> adapterTo = new ArrayAdapter<>(context: TicketActivity.this, android.R.layout.simple_spinner_item, cityNames);
89                     to.setAdapter(adapterTo);
90                     to.setOnItemSelectedListener(TicketActivity.this);
91
92                 }
93             }
94         }
95     );
96
97     jsonObjectRequest.setRetryPolicy(new DefaultRetryPolicy(0, 0, 0));
98     jsonObjectRequest.setShouldCache(true);
99     jsonObjectRequest.setPriority(Request.Priority.HIGH);
100
101     RequestQueue requestQueue = Volley.newRequestQueue(this);
102     requestQueue.add(jsonObjectRequest);
103
104     Intent intent = new Intent(this, MainActivity.class);
105     intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
106     startActivity(intent);
107
108     finish();
109 }
```

```
91             } catch (Exception e) {
92                 Log.e( tag: "Ezgo", msg: "Error: " + e);
93             }
94         }
95     },
96     error -> {
97         Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
98         String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
99         Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
100    });
101 queue.add(jsonObjectRequest);

103 ArrayAdapter<String> adapterType = new ArrayAdapter<>( context: this, android.R.layout.simple_spinner_item, types);
104 type.setAdapter(adapterType);
105 type.setOnItemSelectedListener(this);

107 ArrayAdapter<String> adapterPassanger = new ArrayAdapter<>( context: this, android.R.layout.simple_spinner_item, passangers);
108 passenger.setAdapter(adapterPassanger);
109 passenger.setOnItemSelectedListener(this);

111 calender.setOnClickListener(view -> {
112     Calendar calendar = Calendar.getInstance();
113     tahun = calendar.get(Calendar.YEAR);
114     bulan = calendar.get(Calendar.MONTH);
115     tanggal = calendar.get(Calendar.DAY_OF_MONTH);

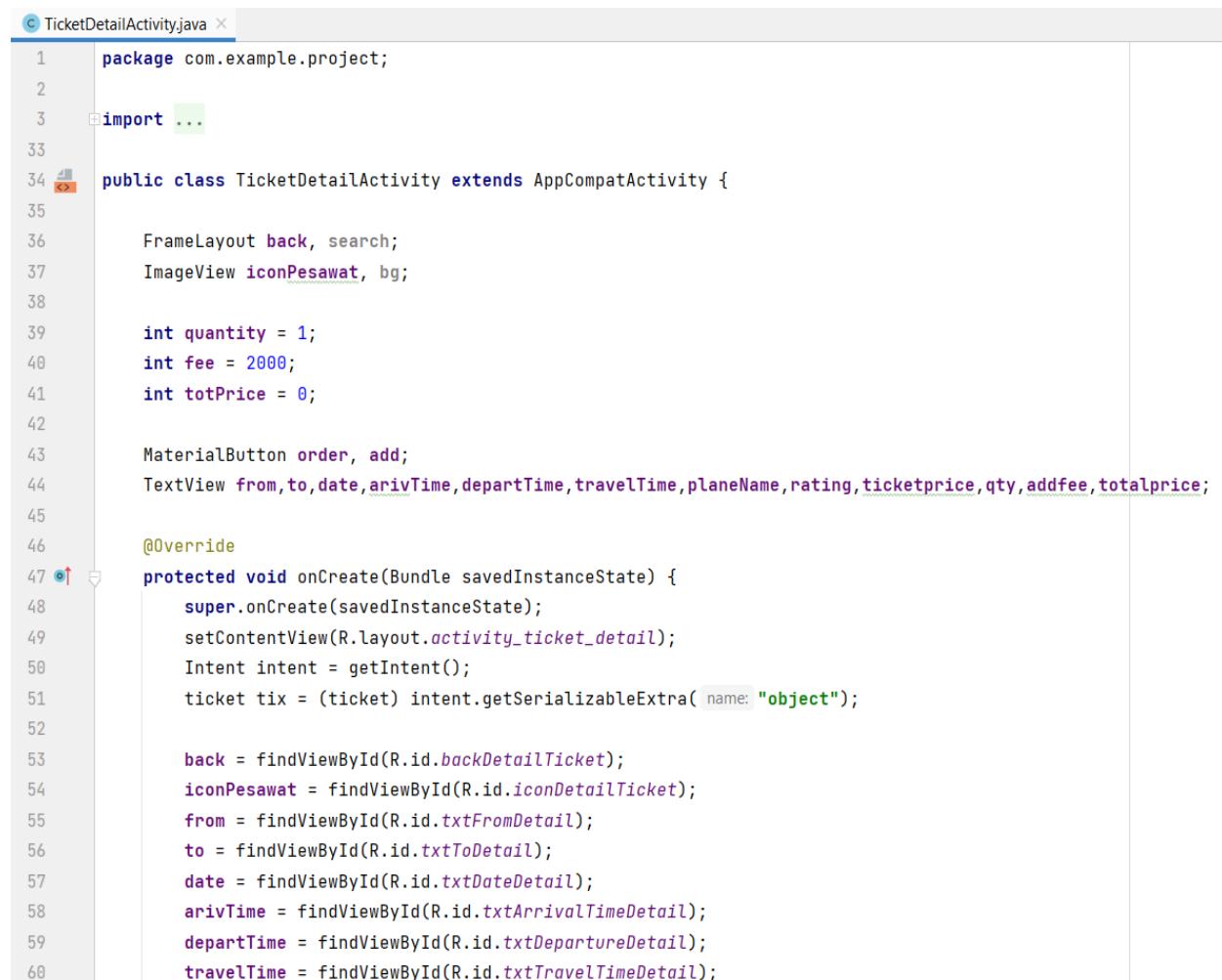
117     DatePickerDialog dialog = new DatePickerDialog( context: TicketActivity.this, (datePicker, i, ii, ii2) -> {
118         tahun = i;
119         bulan = ii + 1;
120         tanggal = ii2;
```

```
 122             calender.setText(String.format("%04d-%02d-%02d", tahun, bulan, tanggal));
 123         }, tahun, bulan, tanggal);
 124         dialog.show();
 125     });
 126 }
 127
 128 //btn
 129 btnPesawat.setOnClickListener(view -> {
 130     int toCity = 0;
 131     int fromCity = 0;
 132
 133     String targetFrom = from.getSelectedItem().toString();
 134     String targetTo = to.getSelectedItem().toString();
 135
 136     for (city obj : cities) {
 137         if (targetFrom.equals(obj.cName)) {
 138             fromCity = obj.cityID;
 139         } else if (targetTo.equals(obj.cName)) {
 140             toCity = obj.cityID;
 141         }
 142     }
 143
 144     String date = calender.getText().toString();
 145     int pas = Integer.parseInt(passenger.getSelectedItem().toString());
 146     String ticktype = type.getSelectedItem().toString();
 147
 148     Intent i = new Intent(getApplicationContext(), TicketViewActivity.class);
 149     i.putExtra( name: "from", fromCity);
 150     i.putExtra( name: "to", toCity);
 151     i.putExtra( name: "fromStr", from.getSelectedItem().toString());
 152     i.putExtra( name: "toStr", to.getSelectedItem().toString());
 153
 154     i.putExtra( name: "date", date);
 155     i.putExtra( name: "pas", pas);
 156     i.putExtra( name: "type", ticktype);
 157     startActivity(i);
 158 }
 159
 160     back.setOnClickListener(view -> onBackPressed());
 161     search.setOnClickListener(view -> {
 162         Intent i = new Intent(getApplicationContext(), SearchActivity.class);
 163         startActivity(i);
 164     });
 165
 166     @Override
 167     public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
 168
 169     }
 170
 171     @Override
 172     public void onNothingSelected(AdapterView<?> adapterView) {
 173
 174     }
 175 }
```

```
 153     i.putExtra( name: "date", date);
 154     i.putExtra( name: "pas", pas);
 155     i.putExtra( name: "type", ticktype);
 156     startActivity(i);
 157 }
 158
 159     back.setOnClickListener(view -> onBackPressed());
 160     search.setOnClickListener(view -> {
 161         Intent i = new Intent(getApplicationContext(), SearchActivity.class);
 162         startActivity(i);
 163     });
 164 }
 165
 166     @Override
 167     public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
 168
 169     }
 170
 171     @Override
 172     public void onNothingSelected(AdapterView<?> adapterView) {
 173
 174     }
 175 }
```

Kode java TicketActivity di atas merupakan sebuah kelas dalam aplikasi Android yang berfungsi sebagai tampilan untuk pencarian tiket perjalanan. Aktivitas ini memungkinkan pengguna untuk memilih jenis transportasi (pesawat, kereta, bus), jumlah penumpang, kota asal, kota tujuan, dan tanggal keberangkatan melalui beberapa komponen antarmuka seperti *Spinner* dan *DatePickerDialog*. Data kota diambil dari server melalui permintaan HTTP menggunakan library *Volley* dan ditampilkan dalam *spinner*. Setelah pengguna mengisi semua informasi, mereka dapat menekan tombol pencarian untuk melanjutkan ke aktivitas lain (*TicketViewActivity*) yang akan menampilkan hasil pencarian berdasarkan parameter yang telah dipilih. Kode ini juga menyediakan fungsi navigasi kembali ke halaman sebelumnya dan pencarian global dalam aplikasi.

### **3.2.49. TicketDetailActivity**



```
1 package com.example.project;
2
3 import ...
4
5
6 public class TicketDetailActivity extends AppCompatActivity {
7
8     FrameLayout back, search;
9     ImageView iconPesawat, bg;
10
11     int quantity = 1;
12     int fee = 2000;
13     int totPrice = 0;
14
15     MaterialButton order, add;
16     TextView from,to,date,arivTime,departTime,travelTime,planeName,rating,ticketprice,qty,addfee,totalprice;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_ticket_detail);
22         Intent intent = getIntent();
23         ticket tix = (ticket) intent.getSerializableExtra( name: "object");
24
25         back = findViewById(R.id.backDetailTicket);
26         iconPesawat = findViewById(R.id.iconDetailTicket);
27         from = findViewById(R.id.txtFromDetail);
28         to = findViewById(R.id.txtToDetail);
29         date = findViewById(R.id.txtDateDetail);
30         arivTime = findViewById(R.id.txtArrivalTimeDetail);
31         departTime = findViewById(R.id.txtDepartureDetail);
32         travelTime = findViewById(R.id.txtTravelTimeDetail);
33     }
34 }
```

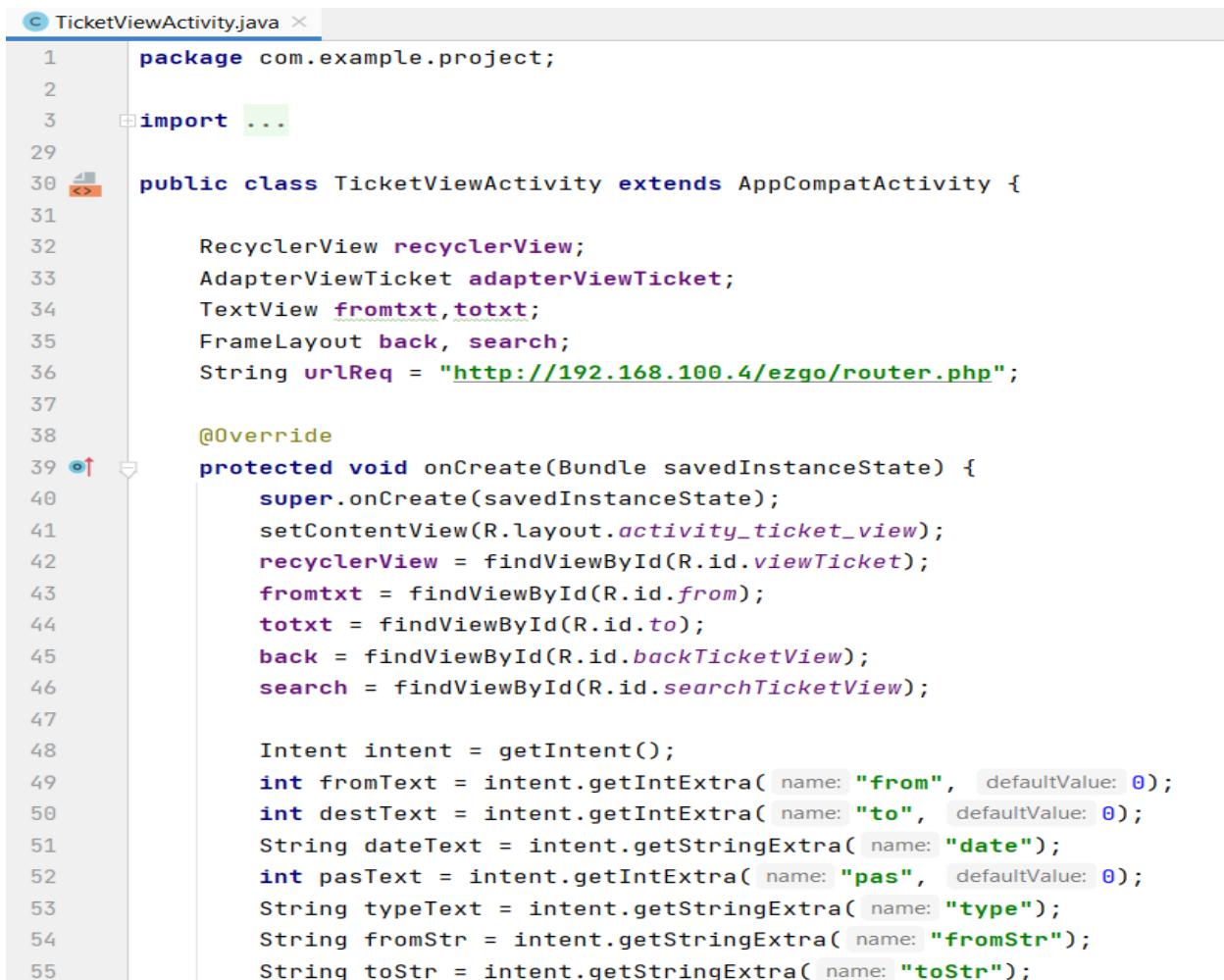
```
  C TicketDetailActivity.java x
11   planeName = findViewById(R.id.txtNamaDetail);
12   rating = findViewById(R.id.txtRatingTicket);
13   ticketprice = findViewById(R.id.ticketPrice);
14   qty = findViewById(R.id.ticketQty);
15   addfee = findViewById(R.id.addFee);
16   totalprice = findViewById(R.id.totalPrice);
17   order = findViewById(R.id.orderList);
18   add = findViewById(R.id.addMore);
19
20
21   DecimalFormat decimalFormat = new DecimalFormat( pattern: "#,###");
22
23   totPrice = tix.tcPrice * quantity + fee;
24
25   String urlReq = "http://192.168.100.4/ezgo/router.php";
26   String urlImg = "http://192.168.100.4/ezgo/images/";
27
28   RequestQueue queue = Volley.newRequestQueue( context: this);
29   Gson gson = new Gson();
30
31   Map<String, Object> params = new HashMap<>();
32   params.put( k: "controller", v: "city");
33   params.put( k: "method", v: "getCityTwo");
34   params.put( k: "cityIDfrom", tix.tcFrom);
35   params.put( k: "cityIDdest", tix.tcDestination);
36
37   JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
38     new JSONObject(params),
39     response -> {
40       Log.d( tag: "Ezgo", msg: "ResponseHome: " + response);
41       ResponseOneObjectList<String> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObjectList<String>() {}.getType());
42       boolean success = resp.isSuccess();
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
```

```
  C TicketDetailActivity.java x
92   List<String> cNames = resp.getData();
93
94   if (success) {
95     try {
96       from.setText(cNames.get(0));
97       to.setText(cNames.get(1));
98       travelTime.setText(tix.tcTravelTime);
99       departTime.setText(tix.tcDepartureTime);
100
101      DateTimeFormatter formatter = null;
102      if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
103        formatter = DateTimeFormatter.ofPattern("HH:mm:ss");
104      }
105
106      LocalTime sum = null;
107      if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
108        LocalTime time1 = LocalTime.parse(tix.tcDepartureTime, formatter);
109        LocalTime time2 = LocalTime.parse(tix.tcTravelTime, formatter);
110
111        sum = time1.plusHours(time2.getHour())
112          .plusMinutes(time2.getMinute())
113          .plusSeconds(time2.getSecond());
114      }
115
116
117      SimpleDateFormat inputFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd");
118      Date dateForm = inputFormat.parse(tix.tcDate);
119      SimpleDateFormat outputFormat = new SimpleDateFormat( pattern: "dd MMMM yyyy");
120      String outputDateString = outputFormat.format(dateForm);
121
122      date.setText(outputDateString);
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
```

```
c TicketDetailActivity.java x
123     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
124         arivTime.setText(sum.format(formatter));
125     }
126     rating.setText(Double.toString(tix.tcRating));
127     ticketprice.setText(decimalFormat.format(tix.tcPrice));
128     qty.setText(Integer.toString(quantity));
129     addfee.setText(decimalFormat.format(fee));
130     totalprice.setText(decimalFormat.format(totPrice));
131 }catch (Exception e){
132     Log.e( tag: "Ezgo", msg: "Error: " + e);
133 }
134 }
135 },
136 error -> {
137     Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
138     String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
139     Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
140 });
141 queue.add(jsonObjectRequest);
142
143 back.setOnClickListener(view -> onBackPressed());
144
145 add.setOnClickListener(view -> {
146     if((quantity + 1) <= tix.tcSeat){
147         quantity++;
148         totPrice = tix.tcPrice * quantity + fee;
149         qty.setText(Integer.toString(quantity));
150         totalprice.setText(decimalFormat.format(totPrice));
151     }else{
152         Toast.makeText(getApplicationContext(), text: "Max Amount Reached", Toast.LENGTH_LONG).show();
153     }
154 });
155
156 order.setOnClickListener(view -> {
157     Intent i = new Intent(getApplicationContext(), PaymentActivity.class);
158     i.putExtra( name: "object", tix);
159     i.putExtra( name: "price", totPrice);
160     i.putExtra( name: "amount", quantity);
161     startActivity(i);
162 });
163
164 }
165 }
```

Kode java TicketDetailActivity di atas merupakan sebuah kelas dalam aplikasi Android yang menampilkan detail tiket perjalanan yang dipilih oleh pengguna. Aktivitas ini mengambil data tiket dari *Intent* yang mengirimkan objek *ticket* dan menampilkan informasi seperti asal, tujuan, tanggal, waktu keberangkatan, waktu tiba, waktu perjalanan, nama pesawat, rating, harga tiket, dan jumlah tiket. Pengguna dapat menambah jumlah tiket yang ingin dipesan dengan tombol “*add*”, yang akan memperbarui total harga yang harus dibayar. Jika pengguna mengklik tombol “*order*”, aplikasi akan berpindah ke aktivitas *PaymentActivity* untuk memproses pembayaran. Aktivitas ini juga mengambil data nama kota asal dan tujuan dari server menggunakan permintaan HTTP dengan library *Volley*, serta mengonversi dan menampilkan data tanggal dan waktu dengan format yang mudah dibaca.

### **3.2.50. TicketViewActivity**



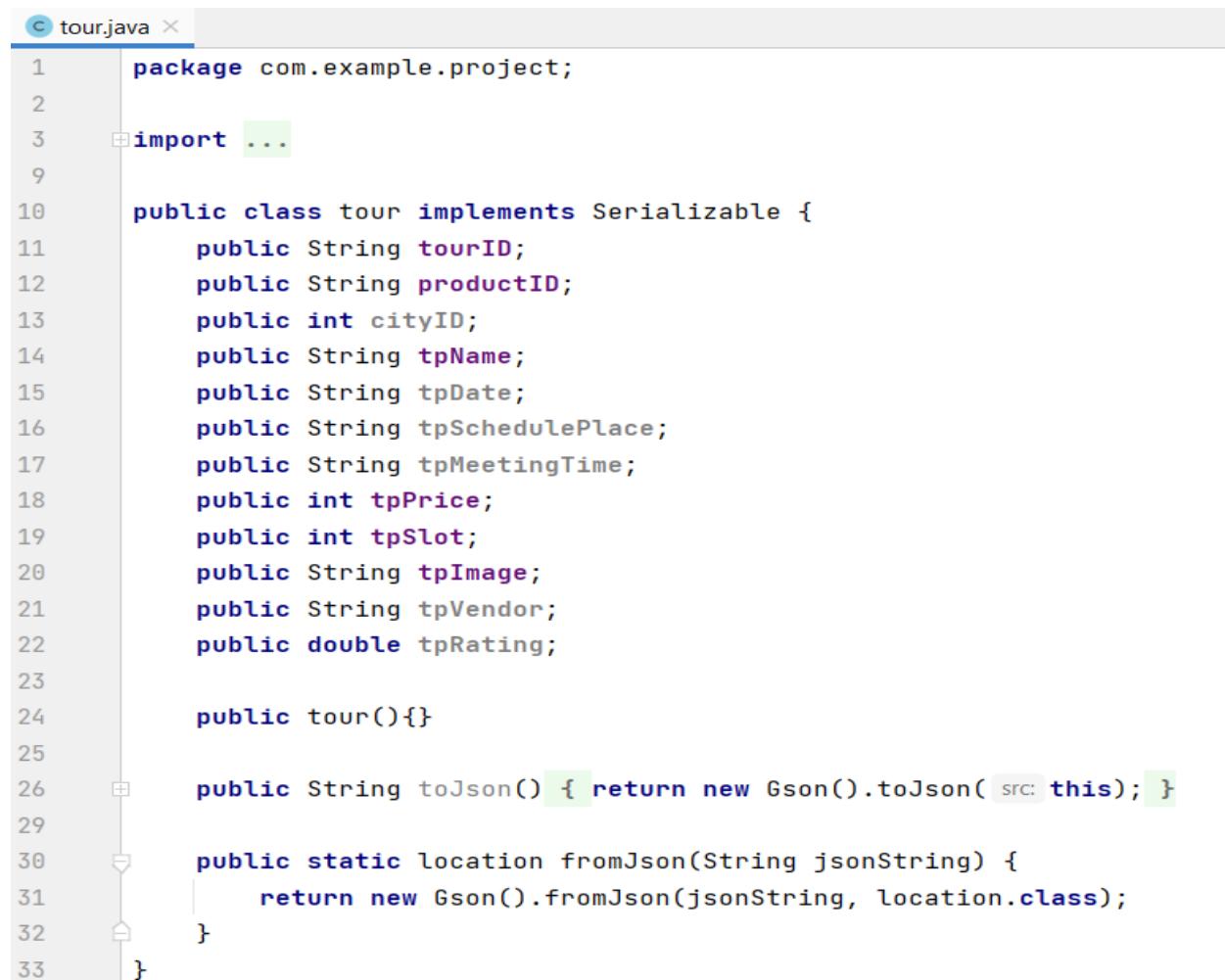
```
1 package com.example.project;
2
3 import ...
29
30 public class TicketViewActivity extends AppCompatActivity {
31
32     RecyclerView recyclerView;
33     AdapterViewTicket adapterViewTicket;
34     TextView fromtxt, totxt;
35     FrameLayout back, search;
36     String urlReq = "http://192.168.100.4/ezgo/router.php";
37
38     @Override
39     protected void onCreate(Bundle savedInstanceState) {
40         super.onCreate(savedInstanceState);
41         setContentView(R.layout.activity_ticket_view);
42         recyclerView = findViewById(R.id.viewTicket);
43         fromtxt = findViewById(R.id.from);
44         totxt = findViewById(R.id.to);
45         back = findViewById(R.id.backTicketView);
46         search = findViewById(R.id.searchTicketView);
47
48         Intent intent = getIntent();
49         int fromText = intent.getIntExtra("from", 0);
50         int destText = intent.getIntExtra("to", 0);
51         String dateText = intent.getStringExtra("date");
52         int pasText = intent.getIntExtra("pas", 0);
53         String typeText = intent.getStringExtra("type");
54         String fromStr = intent.getStringExtra("fromStr");
55         String toStr = intent.getStringExtra("toStr");
```

```
57 Log.d( tag: "Ezgo", msg: "Params: " + fromText + destText + dateText + pasText + typeText);
58
59 RequestQueue queue = Volley.newRequestQueue( context: this);
60 Gson gson = new Gson();
61
62 Map<String, Object> params = new HashMap<>();
63 params.put( k: "controller", v: "order");
64 params.put( k: "method", v: "searchTicket");
65 params.put( k: "tcFrom", fromText);
66 params.put( k: "tcDestination", destText);
67 params.put( k: "tcDate", dateText);
68 params.put( k: "tcSeat", pasText);
69 params.put( k: "tcType", typeText);
70
71 JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
72     new JSONObject(params),
73     new Response.Listener<JSONObject>() {
74         @Override
75         public void onResponse(JSONObject response) {
76             Log.d( tag: "Ezgo", msg: "ResponseHome: " + response);
77             ResponseOneObjectList<ticket> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObjectList<ticket>>() {}.getType());
78             boolean success = resp.isSuccess();
79             List<ticket> tickets = resp.getData();
80
81             if (success == true) {
82                 try {
83                     adapterViewTicket = new AdapterViewTicket( context: TicketViewActivity.this, tickets);
84                     recyclerView.setLayoutManager(new LinearLayoutManager( context: TicketViewActivity.this));
85                     recyclerView.setAdapter(adapterViewTicket);
86                 }catch (Exception e){
87                     Log.e( tag: "Ezgo", msg: "Error: " + e);
88                 }
89             }
90         }
91     });
92
93     jsonObjectRequest.setRetryPolicy(new DefaultRetryPolicy(0, 0, 0));
94     queue.add(jsonObjectRequest);
95 }
```

```
88
89
90
91
92     new Response.ErrorListener() {
93
94     @Override
95     public void onErrorResponse(VolleyError error) {
96
97         Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
98
99         String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
100
101         Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
102
103     });
104
105     queue.add(jsonObjectRequest);
106
107     fromtxt.setText(fromStr);
108     totxt.setText(toStr);
109
110     back.setOnClickListener(view -> onBackPressed());
111     search.setOnClickListener(view -> startActivity(new Intent(getApplicationContext(), SearchActivity.class)));
112
113 }
```

Kode java TicketViewActivity di atas merupakan sebuah kelas dalam aplikasi Android yang menampilkan hasil pencarian tiket berdasarkan kriteria yang diinput oleh pengguna. Aktivitas ini menerima data dari aktivitas sebelumnya melalui *Intent*, seperti kota asal, kota tujuan, tanggal perjalanan, jumlah penumpang, dan jenis transportasi. Kemudian, aktivitas ini mengirimkan permintaan HTTP *POST* ke server untuk mencari tiket yang sesuai dengan kriteria tersebut menggunakan library *Volley*. Data tiket yang diperoleh dari server kemudian ditampilkan dalam *RecyclerView* menggunakan adapter *AdapterViewTicket*. Aktivitas ini juga menyediakan navigasi untuk kembali ke halaman sebelumnya dan untuk membuka halaman pencarian. Kode ini mengimplementasikan fitur pencarian tiket, pengambilan data dari server, dan menampilkan data tiket dalam format yang mudah dibaca pengguna.

### 3.2.51. tour



```
1 package com.example.project;
2
3 import ...
4
5
6 public class tour implements Serializable {
7     public String tourID;
8     public String productID;
9     public int cityID;
10    public String tpName;
11    public String tpDate;
12    public String tpSchedulePlace;
13    public String tpMeetingTime;
14    public int tpPrice;
15    public int tpSlot;
16    public String tpImage;
17    public String tpVendor;
18    public double tpRating;
19
20    public tour(){}
21
22    public String toJson() { return new Gson().toJson( src: this); }
23
24    public static location fromJson(String jsonString) {
25        return new Gson().fromJson(jsonString, location.class);
26    }
27}
```

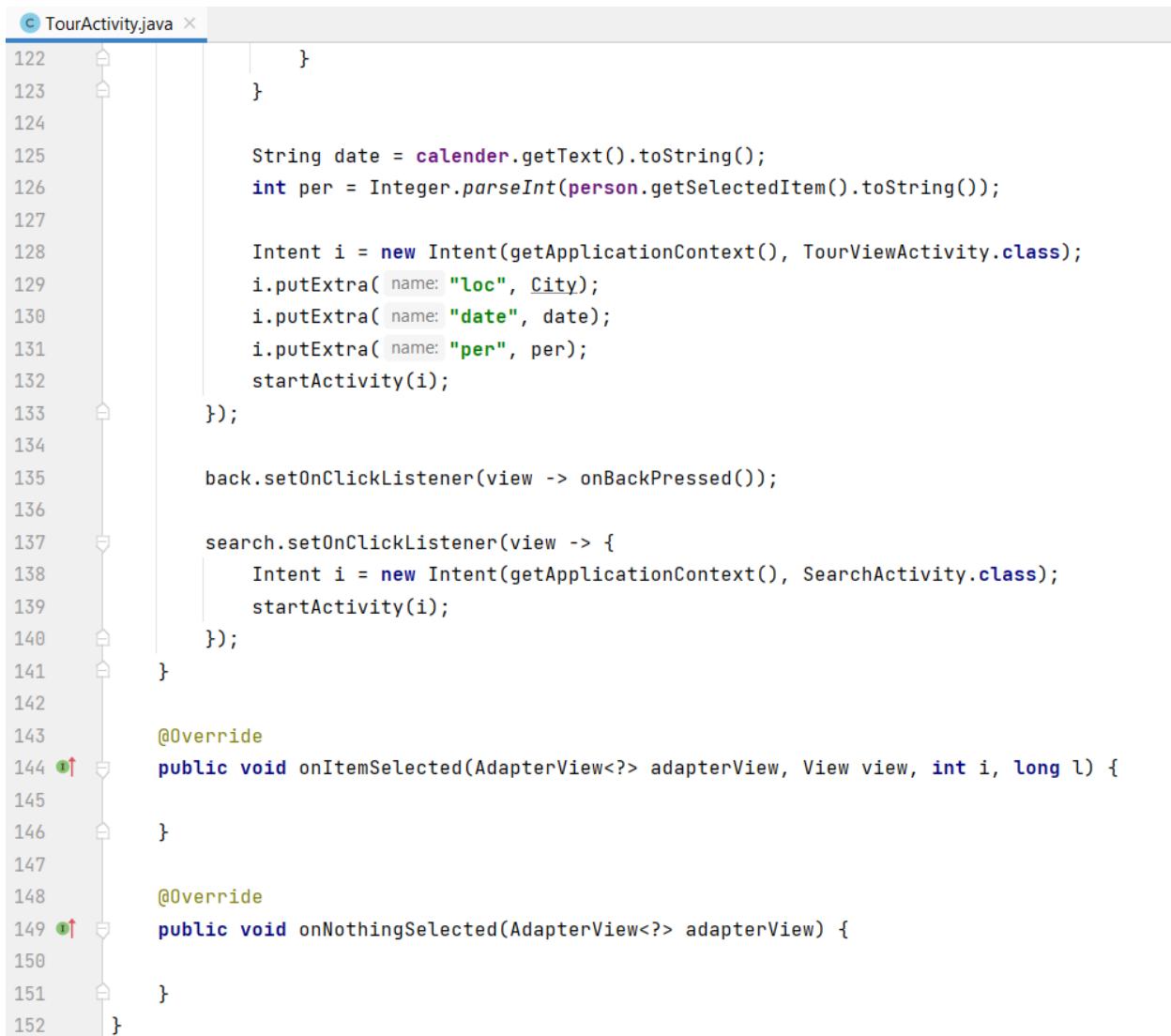
Kode java tour di atas merupakan sebuah kelas model dalam aplikasi Android yang mewakili data sebuah paket tour. Kelas ini mengimplementasikan antarmuka *Serializable* sehingga objek-objeknya dapat dengan mudah dikirimkan antar aktivitas atau disimpan dalam file. Setiap tour memiliki atribut seperti *tourID*, *productID*, *cityID*, *tpName* (nama tour), *tpDate* (tanggal tour), *tpSchedulePlace* (tempat jadwal), *tpMeetingTime* (waktu pertemuan), *tpPrice* (harga), *tpSlot* (slot tersedia), *tpImage* (gambar), *tpVendor* (vendor), dan *tpRating* (rating). Kelas ini juga menyediakan metode *toJson* untuk mengubah objek tour menjadi JSON string dan metode statis *fromJson* untuk mengubah JSON string kembali menjadi objek *location*. Kode ini berguna untuk memudahkan pengelolaan data paket tour dalam aplikasi, seperti menyimpan, mengirim, dan menampilkan informasi tour.

### 3.2.52. *TourActivity*

```
1 package com.example.project;
2
3 import ...
4
5
6 public class TourActivity extends AppCompatActivity implements AdapterView.OnItemSelectedListener {
7     private int tanggal, bulan, tahun;
8     private TextView calender;
9     private Spinner person, location;
10    private List<city> cities;
11    private String urlReq = "http://192.168.100.4/ezgo/router.php";
12
13    private final String[] number = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10"};
14
15    @Override
16    protected void onCreate(Bundle savedInstanceState) {
17        super.onCreate(savedInstanceState);
18        setContentView(R.layout.activity_tour);
19        calender = findViewById(R.id.calender);
20        person = findViewById(R.id.slotTourSpin);
21        location = findViewById(R.id.cityTour);
22        FrameLayout back = findViewById(R.id.backTour);
23        FrameLayout search = findViewById(R.id.searchTour);
24        MaterialButton btnTour = findViewById(R.id.btnSearchTour);
25
26        RequestQueue queue = Volley.newRequestQueue( context: this );
27        Gson gson = new Gson();
28
29        Map<String, Object> params = new HashMap<>();
30        params.put( k: "controller", v: "city" );
31        params.put( k: "method", v: "list" );
32    }
33}
```

```
60     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
61         new JSONObject(params),
62         response -> {
63             Log.d( tag: "Ezgo", msg: "ResponseHome: " + response);
64             ResponseOneObjectList<city> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObjectList<city>>() {
65                 .getType());
66             boolean success = resp.isSuccess();
67             cities = resp.getData();
68
69             if (success == true) {
70                 try {
71                     List<String> cityNames = new ArrayList<>();
72
73                     for (city city : cities) {
74                         cityNames.add(city.cName);
75                         Log.d( tag: "Ezgo", msg: "ResponseHome: " + cityNames);
76                     }
77
78                     ArrayAdapter<String> adapterFrom = new ArrayAdapter<>( context: TourActivity.this, android.R.layout.simple_spinner_item, cityNames);
79                     location.setAdapter(adapterFrom);
80                     location.setOnItemSelectedListener(TourActivity.this);
81                 } catch (Exception e) {
82                     Log.e( tag: "Ezgo", msg: "Error: " + e);
83                 }
84             },
85         },
86         error -> {
87             Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
88             String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
89             Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
90         });
91     });
92 }
```

```
91     queue.add(jsonObjectRequest);
92
93     ArrayAdapter<String> adapterPerson = new ArrayAdapter<>(context: this, android.R.layout.simple_spinner_item, number);
94     person.setAdapter(adapterPerson);
95     person.setOnItemSelectedListener(this);
96
97     calender.setOnClickListener(view -> {
98         Calendar calendar = Calendar.getInstance();
99         tahun = calendar.get(Calendar.YEAR);
100        bulan = calendar.get(Calendar.MONTH);
101        tanggal = calendar.get(Calendar.DAY_OF_MONTH);
102
103        DatePickerDialog dialog = new DatePickerDialog(context: TourActivity.this, (datePicker, i, i1, i2) -> {
104            tahun = i;
105            bulan = i1 + 1;
106            tanggal = i2;
107
108            calender.setText(String.format("%04d-%02d-%02d", tahun, bulan, tanggal));
109        }, tahun, bulan, tanggal);
110        dialog.show();
111
112    });
113
114    btnTour.setOnClickListener(view -> {
115        int City = 0;
116
117        String targetCity = location.getSelectedItem().toString();
118
119        for (city obj : cities) {
120            if (targetCity.equals(obj.cName)) {
121                City = obj.cityID;
```



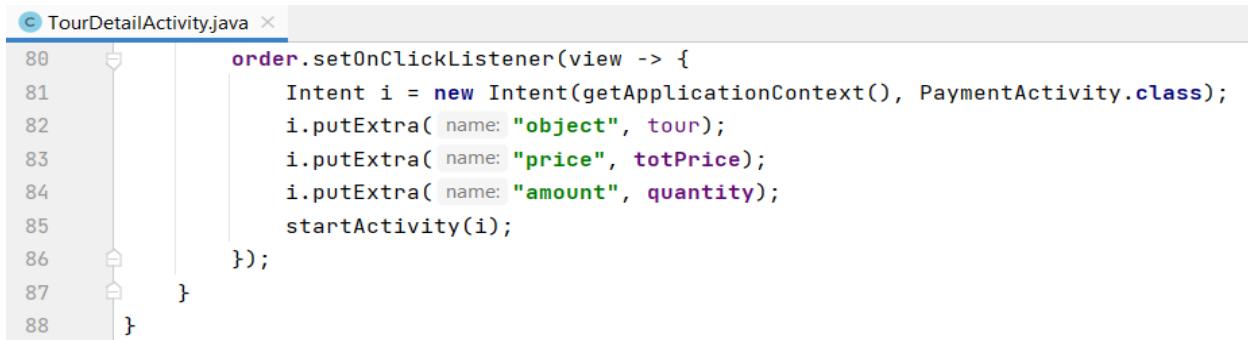
```
122     }
123     }
124
125     String date = calendar.getText().toString();
126     int per = Integer.parseInt(person.getSelectedItem().toString());
127
128     Intent i = new Intent(getApplicationContext(), TourViewActivity.class);
129     i.putExtra( name: "loc", City);
130     i.putExtra( name: "date", date);
131     i.putExtra( name: "per", per);
132     startActivity(i);
133 );
134
135     back.setOnClickListener(view -> onBackPressed());
136
137     search.setOnClickListener(view -> {
138         Intent i = new Intent(getApplicationContext(), SearchActivity.class);
139         startActivity(i);
140     );
141 }
142
143     @Override
144     public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
145
146     }
147
148     @Override
149     public void onNothingSelected(AdapterView<?> adapterView) {
150
151     }
152 }
```

Kode java TourActivity di atas merupakan sebuah aktivitas dalam aplikasi Android yang memungkinkan pengguna untuk mencari paket tour berdasarkan lokasi, tanggal, dan jumlah orang. Dalam aktivitas ini, pengguna dapat memilih kota tujuan, tanggal tour, dan jumlah orang yang akan ikut serta dalam tour. Data kota diambil dari server menggunakan permintaan HTTP dengan menggunakan library *Volley* dan ditampilkan dalam *Spinner*. Tanggal tour dipilih menggunakan *DatePickerDialog*. Setelah pengguna mengisi semua pilihan, tombol “*Search Tour*” akan mengarahkan pengguna ke *TourViewActivity* dengan parameter kota, tanggal, dan jumlah orang yang telah dipilih. Fitur-fitur yang diimplementasikan dalam kode ini meliputi pengambilan data dari server, pengisian *Spinner*, penggunaan *DatePickerDialog*, dan pengiriman data antar aktivitas.

### 3.2.53. TourDetailActivity

```
1 package com.example.project;
2
3 import ...
4
5
6 public class TourDetailActivity extends AppCompatActivity{
7     FrameLayout back;
8     ImageView img;
9     TextView name, price, nights, totprice, addfee;
10    MaterialButton add, order;
11
12    int quantity = 1;
13    int fee = 2000;
14    int totPrice = 0;
15
16    @Override
17    protected void onCreate(Bundle savedInstanceState) {
18        super.onCreate(savedInstanceState);
19        setContentView(R.layout.activity_tour_detail);
20        Intent intent = getIntent();
21        tour tour = (tour) intent.getSerializableExtra("name: "object");
22
23        back = findViewById(R.id.backDetailTour);
24        img = findViewById(R.id.imgTour);
25        name = findViewById(R.id.htlName);
26        price = findViewById(R.id.priceTour);
27        nights = findViewById(R.id.slotTour);
28        totprice = findViewById(R.id.totPriceTour);
29        addfee = findViewById(R.id.addFeeTour);
30        add = findViewById(R.id.addMore);
31        order = findViewById(R.id.orderTour);
32    }
33
34    public void addMore() {
35        quantity++;
36        totPrice += fee;
37        name.setText("Name: " + quantity);
38        price.setText("Price: " + totPrice);
39    }
40
41    public void order() {
42        Intent intent = new Intent(this, OrderActivity.class);
43        intent.putExtra("name: "object", tour);
44        intent.putExtra("totPrice: "object", totPrice);
45        intent.putExtra("quantity: "object", quantity);
46        startActivity(intent);
47    }
48
49    public void back() {
50        finish();
51    }
52
53    public void img() {
54        Intent intent = new Intent(this, ImageActivity.class);
55        intent.putExtra("img: "object", img);
56        startActivity(intent);
57    }
58
59    public void name() {
60        Intent intent = new Intent(this, NameActivity.class);
61        intent.putExtra("name: "object", name);
62        startActivity(intent);
63    }
64
65    public void price() {
66        Intent intent = new Intent(this, PriceActivity.class);
67        intent.putExtra("price: "object", price);
68        startActivity(intent);
69    }
70
71    public void nights() {
72        Intent intent = new Intent(this, NightsActivity.class);
73        intent.putExtra("nights: "object", nights);
74        startActivity(intent);
75    }
76
77    public void totprice() {
78        Intent intent = new Intent(this, TotPriceActivity.class);
79        intent.putExtra("totprice: "object", totprice);
80        startActivity(intent);
81    }
82
83    public void addfee() {
84        Intent intent = new Intent(this, AddFeeActivity.class);
85        intent.putExtra("addfee: "object", addfee);
86        startActivity(intent);
87    }
88
89    public void add() {
90        Intent intent = new Intent(this, AddActivity.class);
91        intent.putExtra("add: "object", add);
92        startActivity(intent);
93    }
94
95    public void order() {
96        Intent intent = new Intent(this, OrderActivity.class);
97        intent.putExtra("order: "object", order);
98        startActivity(intent);
99    }
100}
```

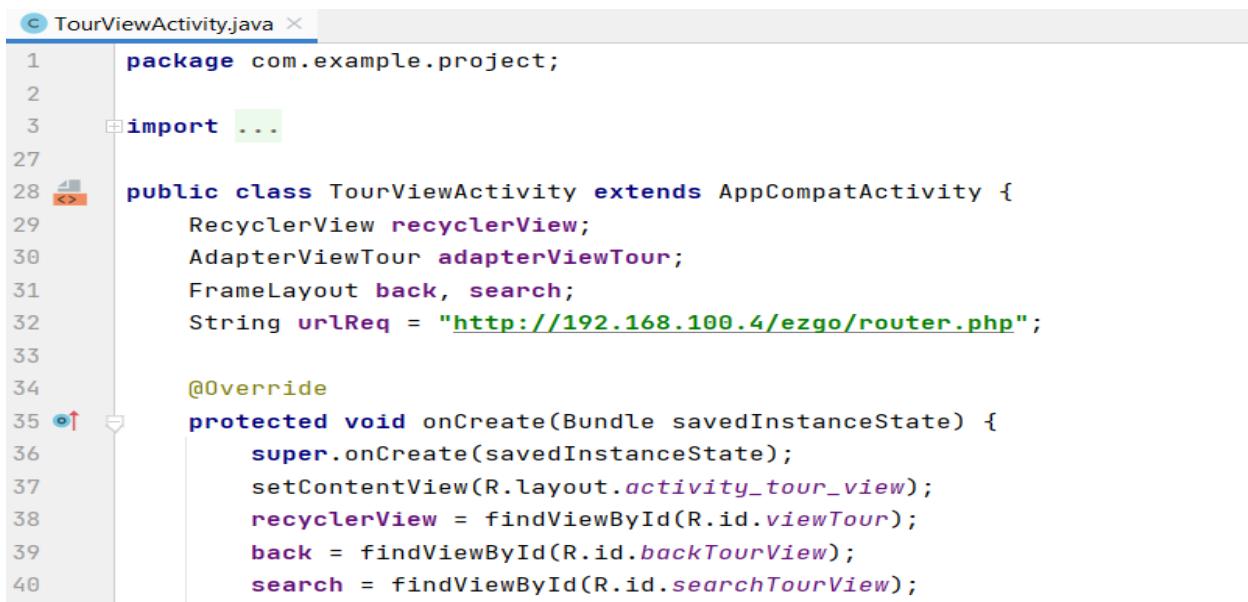
```
52     DecimalFormat decimalFormat = new DecimalFormat( pattern: "#,###");
53
54     totPrice = tour.tpPrice * quantity + fee;
55
56     String urlImg = "http://192.168.100.4/ezgo/images/";
57     String image = urlImg + tour.tpImage;
58
59     Picasso.get().load(image).into(img);
60
61     name.setText(tour.tpName);
62     price.setText(decimalFormat.format(tour.tpPrice));
63     nights.setText(Integer.toString(quantity));
64     addfee.setText(decimalFormat.format(fee));
65     totprice.setText(decimalFormat.format(totPrice));
66
67     back.setOnClickListener(view -> onBackPressed());
68
69     add.setOnClickListener(view -> {
70         if((quantity + 1) <= tour.tpSlot){
71             quantity++;
72             totPrice = tour.tpPrice * quantity + fee;
73             nights.setText(Integer.toString(quantity));
74             totprice.setText(decimalFormat.format(totPrice));
75         }else{
76             Toast.makeText(getApplicationContext(), text: "Max Amount Reached", Toast.LENGTH_LONG).show();
77         }
78     });
});
```



```
80     order.setOnClickListener(view -> {
81         Intent i = new Intent(getApplicationContext(), PaymentActivity.class);
82         i.putExtra( name: "object", tour);
83         i.putExtra( name: "price", totPrice);
84         i.putExtra( name: "amount", quantity);
85         startActivity(i);
86     });
87 }
88 }
```

Kode java TourDetailActivity di atas merupakan aktivitas dalam aplikasi Android yang menampilkan detail sebuah paket tour. Pada aktivitas ini, pengguna dapat melihat informasi mengenai paket tour, seperti nama, harga, dan jumlah slot yang tersedia. Pengguna juga dapat menambah jumlah slot yang ingin dipesan, dan melihat total harga yang akan dibayar. Gambar paket tour diambil dari URL menggunakan library *Picasso*. Fitur utama yang diimplementasikan dalam kode ini adalah pengambilan dan penampilan data detail tour, perhitungan total harga berdasarkan jumlah slot yang dipesan, dan navigasi ke aktivitas pembayaran (*PaymentActivity*). Selain itu, kode ini juga menangani tombol untuk kembali ke halaman sebelumnya dan menambah jumlah slot yang dipesan dengan memastikan jumlah slot tidak melebihi batas yang tersedia.

### 3.2.54. TourViewActivity



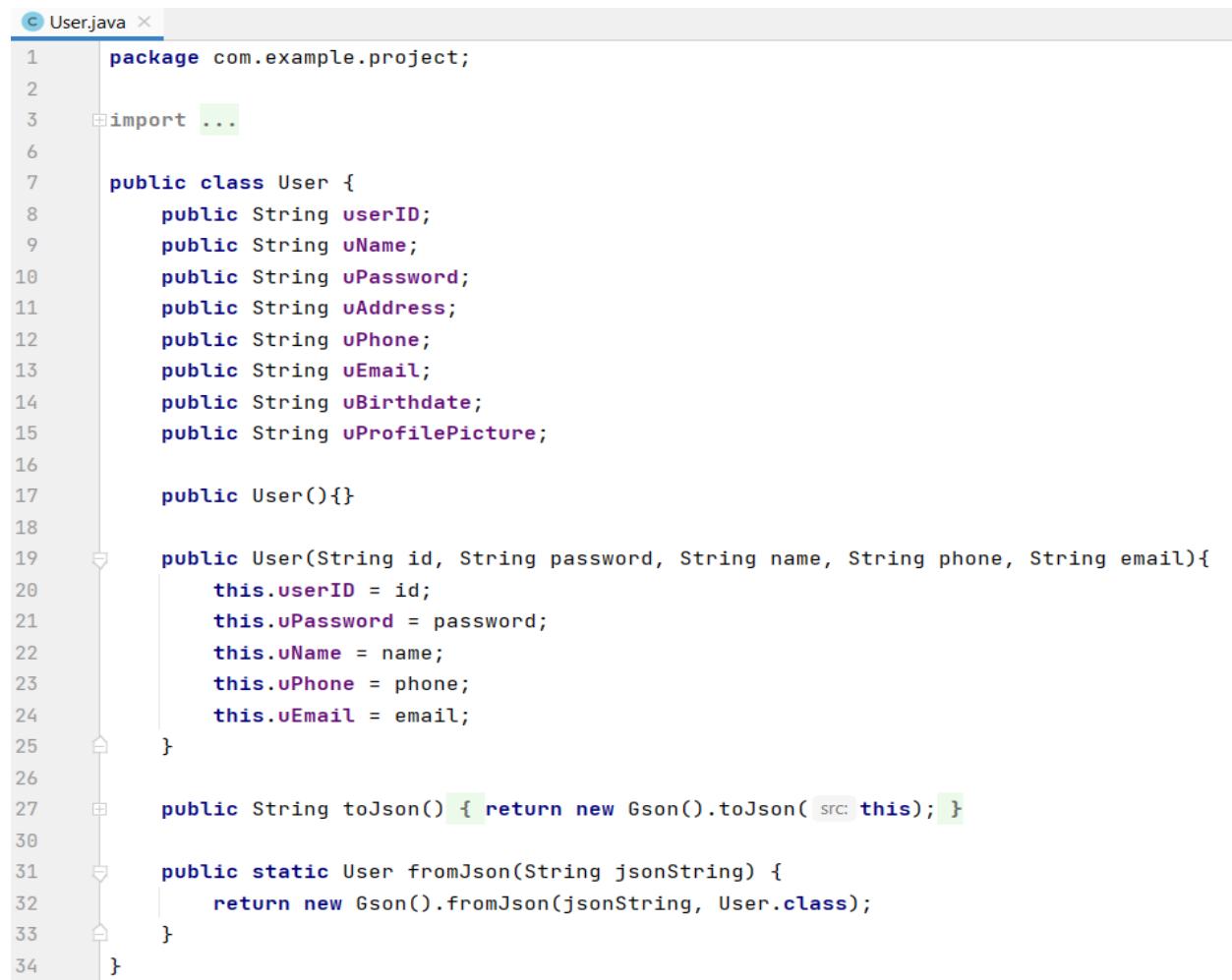
```
1 package com.example.project;
2
3 import ...
27
28 public class TourViewActivity extends AppCompatActivity {
29     RecyclerView recyclerView;
30     AdapterViewTour adapterViewTour;
31     FrameLayout back, search;
32     String urlReq = "http://192.168.100.4/ezgo/router.php";
33
34     @Override
35     protected void onCreate(Bundle savedInstanceState) {
36         super.onCreate(savedInstanceState);
37         setContentView(R.layout.activity_tour_view);
38         recyclerView = findViewById(R.id.viewTour);
39         back = findViewById(R.id.backTourView);
40         search = findViewById(R.id.searchTourView);
```

```
42     Intent intent = getIntent();
43     int cityText = intent.getIntExtra( name: "loc", defaultValue: 0);
44     String dateText = intent.getStringExtra( name: "date");
45     int personText = intent.getIntExtra( name: "per", defaultValue: 0);
46
47     RequestQueue queue = Volley.newRequestQueue( context: this);
48     Gson gson = new Gson();
49
50     Map<String, Object> params = new HashMap<>();
51     params.put( k: "controller", v: "order");
52     params.put( k: "method", v: "searchTour");
53     params.put( k: "cityID", cityText);
54     params.put( k: "tpDate", dateText);
55     params.put( k: "tpSlot", personText);
56
57     JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, urlReq,
58         new JSONObject(params),
59         new Response.Listener<JSONObject>() {
60             @Override
61             public void onResponse(JSONObject response) {
62                 Log.d( tag: "Ezgo", msg: "ResponseHome: " + response);
63                 ResponseOneObjectList<tour> resp = gson.fromJson(response.toString(), new TypeToken<ResponseOneObjectList<tour>>() {}.getType());
64                 boolean success = resp.isSuccess();
65                 List<tour> tourData = resp.getData();
66
67                 if (success == true) {
68                     try {
69                         adapterViewTour = new AdapterViewTour( context: TourViewActivity.this, tourData);
70                         recyclerView.setLayoutManager(new LinearLayoutManager( context: TourViewActivity.this));
71                         recyclerView.setAdapter(adapterViewTour);
72                     }catch (Exception e){
73                         Log.e( tag: "Ezgo", msg: "Error: " + e);
74                     }
75                 }
76             }
77         },
78         new Response.ErrorListener() {
79             @Override
80             public void onErrorResponse(VolleyError error) {
81                 Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
82                 String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
83                 Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
84             }
85         });
86     queue.add(jsonObjectRequest);
87
88     back.setOnClickListener(view -> onBackPressed());
89     search.setOnClickListener(view -> startActivity(new Intent(getApplicationContext(), SearchActivity.class)));
90
91 }
```

```
73             }
74         }
75     }
76 }
77 },
78 new Response.ErrorListener() {
79     @Override
80     public void onErrorResponse(VolleyError error) {
81         Log.e( tag: "Ezgo", msg: "Error: " + error.toString());
82         String responseBody = new String(error.networkResponse.data, StandardCharsets.UTF_8);
83         Log.e( tag: "Ezgo", msg: "Response: " + responseBody);
84     }
85 });
86 queue.add(jsonObjectRequest);
87
88 back.setOnClickListener(view -> onBackPressed());
89 search.setOnClickListener(view -> startActivity(new Intent(getApplicationContext(), SearchActivity.class)));
90
91 }
```

Kode java TourViewActivity di atas merupakan aktivitas dalam aplikasi Android yang menampilkan daftar paket tour berdasarkan kriteria pencarian yang telah ditentukan oleh pengguna. Pada saat aktivitas ini dimulai, data kriteria seperti lokasi kota, tanggal tour, dan jumlah orang yang akan mengikuti tour diambil dari *Intent*. Kode ini kemudian mengirimkan permintaan HTTP *POST* menggunakan library *Volley* untuk mendapatkan daftar paket tour yang sesuai dari server. Data yang diterima dari server diolah dan ditampilkan dalam *RecyclerView* menggunakan *AdapterViewTour*. Selain itu, aktivitas ini juga menangani navigasi tombol kembali ke halaman sebelumnya dan tombol pencarian yang membawa pengguna ke *SearchActivity*. Fungsi utama dari kode ini adalah untuk menampilkan daftar tour yang sesuai dengan kriteria pencarian yang diberikan oleh pengguna.

### 3.2.55. User



```
1 package com.example.project;
2
3 import ...
4
5
6
7 public class User {
8     public String userID;
9     public String uName;
10    public String uPassword;
11    public String uAddress;
12    public String uPhone;
13    public String uEmail;
14    public String uBirthdate;
15    public String uProfilePicture;
16
17    public User(){}
18
19    public User(String id, String password, String name, String phone, String email){
20        this.userID = id;
21        this.uPassword = password;
22        this.uName = name;
23        this.uPhone = phone;
24        this.uEmail = email;
25    }
26
27    public String toJson() { return new Gson().toJson( src: this); }
28
29
30
31    public static User fromJson(String jsonString) {
32        return new Gson().fromJson(jsonString, User.class);
33    }
34 }
```

Kode java User di atas mendefinisikan kelas *User* yang digunakan untuk merepresentasikan data pengguna dalam aplikasi. Kelas ini mengimplementasikan *Serializable* sehingga objek *User* dapat diserialisasi. Atribut-atribut yang dimiliki oleh kelas ini mencakup *userID*, *uName*, *uPassword*, *uAddress*, *uPhone*, *uEmail*, *uBirthdate*, dan *uProfilePicture*. Kelas ini memiliki dua *konstruktor*: konstruktor tanpa parameter dan konstruktor dengan parameter untuk inisialisasi beberapa atribut. Kode ini juga mengimplementasikan metode *toJson()* untuk mengonversi objek *User* menjadi format JSON dan metode *fromJson()* untuk mengonversi string JSON kembali menjadi objek *User*, menggunakan library Gson. Fungsi utama dari kode ini adalah untuk mempermudah pengelolaan dan penyimpanan data pengguna dalam aplikasi, serta memungkinkan konversi data ke dan dari format JSON.

### 3.2.56. *WebViewActivity*



```
1 package com.example.project;
2
3 import ...
4
5 public class WebViewActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_web_view);
11        Intent intent = getIntent();
12        String link = intent.getStringExtra(name: "Link");
13
14        WebView wv = (WebView) findViewById(R.id.webview);
15        wv.loadUrl(link);
16    }
17}
```

Kode java WebViewActivity di atas merupakan sebuah activity dalam aplikasi Android yang digunakan untuk menampilkan halaman web di dalam aplikasi. Ketika activity ini dibuat, metode *onCreate()* dipanggil untuk menginisialisasi activity. Dalam metode ini, layout *activity\_web\_view* diatur sebagai tampilan untuk activity ini. *Intent* yang membawa data *link* (URL) diterima dari activity sebelumnya. Sebuah objek *WebView* ditemukan dari layout dan digunakan untuk memuat URL yang diterima melalui metode *loadUrl()*. Dengan demikian, halaman web yang ditentukan oleh URL tersebut akan ditampilkan di dalam aplikasi.

### 3.3. Code Back-End Aplikasi (PHP)

### 3.3.1. accountController.php

```
accountController.php X
C: > xampp > htdocs > ezgo > accountController.php > PHP > account > updateAccount()
1  <?php
2  include_once 'database.php';
3
4  class account{
5      0 references | 0 implementations
6      0 references | 0 overrides
7      public function showAccount(){
8          $db = DB::getInstance();
9          $json = file_get_contents('php://input');
10
11         $request = json_decode($json, true);
12         $uid = $request['userID'];
13
14         $where = array('userID', '=', $uid);
15         $user = $db->select('*', 'users', $where);
16
17         if(count($user) > 0){
18             $imgPath = $user[0]->uProfilePicture;
19             $fullPath = "images/".$imgPath;
20             $user[0]->uProfilePicture = $fullPath;
21
22             $response = ['Success'=> True, 'user' => $user[0]];
23             header('Content-Type: application/json');
24             echo json_encode($response);
25         }else{
26             $response = ['Success'=> False];
27             header('Content-Type: application/json');
28             echo json_encode($response);
29         }
30     }
31
32     0 references | 0 overrides
33     public function updateAccount(){
34         $db = DB::getInstance();
35         $json = file_get_contents('php://input');
36
37         $request = json_decode($json, true);
38         $uid = $request['userID'];
39         $name = $request['uName'];
40         $address = $request['uAddress'];
41         $phone = $request['uPhone'];
42         $email = $request['uEmail'];
43         $bday = $request['uBirthdate'];
44         $data = $request['image'];
45
46         $file_name = basename($data['image']['name']);
47
48         $query = "UPDATE users SET uName = ?, uAddress = ?, uPhone = ?, uEmail = ?, uBirthdate = ? WHERE userID = ?";
49         $stmt = $db->prepare($query);
50         $stmt->execute([$name, $address, $phone, $email, $bday, $uid]);
51
52         $response = ['Success'=> True, 'user' => $user];
53         header('Content-Type: application/json');
54         echo json_encode($response);
55     }
56
57     0 references | 0 overrides
58     public function deleteAccount(){
59         $db = DB::getInstance();
60         $json = file_get_contents('php://input');
61
62         $request = json_decode($json, true);
63         $uid = $request['userID'];
64
65         $query = "DELETE FROM users WHERE userID = ?";
66         $stmt = $db->prepare($query);
67         $stmt->execute([$uid]);
68
69         $response = ['Success'=> True];
70         header('Content-Type: application/json');
71         echo json_encode($response);
72     }
73
74     0 references | 0 overrides
75     public function login(){
76         $db = DB::getInstance();
77         $json = file_get_contents('php://input');
78
79         $request = json_decode($json, true);
80         $username = $request['username'];
81         $password = $request['password'];
82
83         $query = "SELECT * FROM users WHERE uName = ? AND uPassword = ?";
84         $stmt = $db->prepare($query);
85         $stmt->execute([$username, $password]);
86
87         $user = $stmt->fetch();
88
89         if($user){
90             $response = ['Success'=> True, 'user' => $user];
91             header('Content-Type: application/json');
92             echo json_encode($response);
93         }else{
94             $response = ['Success'=> False];
95             header('Content-Type: application/json');
96             echo json_encode($response);
97         }
98     }
99
100    0 references | 0 overrides
101    public function register(){
102        $db = DB::getInstance();
103        $json = file_get_contents('php://input');
104
105        $request = json_decode($json, true);
106        $name = $request['uName'];
107        $address = $request['uAddress'];
108        $phone = $request['uPhone'];
109        $email = $request['uEmail'];
110        $bday = $request['uBirthdate'];
111        $password = $request['uPassword'];
112
113        $query = "INSERT INTO users (uName, uAddress, uPhone, uEmail, uBirthdate, uPassword) VALUES (?, ?, ?, ?, ?, ?)";
114        $stmt = $db->prepare($query);
115        $stmt->execute([$name, $address, $phone, $email, $bday, $password]);
116
117        $response = ['Success'=> True];
118        header('Content-Type: application/json');
119        echo json_encode($response);
120    }
121
122    0 references | 0 overrides
123    public function forgotPassword(){
124        $db = DB::getInstance();
125        $json = file_get_contents('php://input');
126
127        $request = json_decode($json, true);
128        $email = $request['uEmail'];
129
130        $query = "SELECT * FROM users WHERE uEmail = ?";
131        $stmt = $db->prepare($query);
132        $stmt->execute([$email]);
133
134        $user = $stmt->fetch();
135
136        if($user){
137            $response = ['Success'=> True, 'user' => $user];
138            header('Content-Type: application/json');
139            echo json_encode($response);
140        }else{
141            $response = ['Success'=> False];
142            header('Content-Type: application/json');
143            echo json_encode($response);
144        }
145    }
146
147    0 references | 0 overrides
148    public function changePassword(){
149        $db = DB::getInstance();
150        $json = file_get_contents('php://input');
151
152        $request = json_decode($json, true);
153        $uid = $request['userID'];
154        $oldPassword = $request['oldPassword'];
155        $newPassword = $request['newPassword'];
156
157        $query = "UPDATE users SET uPassword = ? WHERE userID = ?";
158        $stmt = $db->prepare($query);
159        $stmt->execute([$newPassword, $uid]);
160
161        $response = ['Success'=> True];
162        header('Content-Type: application/json');
163        echo json_encode($response);
164    }
165
166    0 references | 0 overrides
167    public function changeAddress(){
168        $db = DB::getInstance();
169        $json = file_get_contents('php://input');
170
171        $request = json_decode($json, true);
172        $uid = $request['userID'];
173        $address = $request['uAddress'];
174
175        $query = "UPDATE users SET uAddress = ? WHERE userID = ?";
176        $stmt = $db->prepare($query);
177        $stmt->execute([$address, $uid]);
178
179        $response = ['Success'=> True];
180        header('Content-Type: application/json');
181        echo json_encode($response);
182    }
183
184    0 references | 0 overrides
185    public function changePhone(){
186        $db = DB::getInstance();
187        $json = file_get_contents('php://input');
188
189        $request = json_decode($json, true);
190        $uid = $request['userID'];
191        $phone = $request['uPhone'];
192
193        $query = "UPDATE users SET uPhone = ? WHERE userID = ?";
194        $stmt = $db->prepare($query);
195        $stmt->execute([$phone, $uid]);
196
197        $response = ['Success'=> True];
198        header('Content-Type: application/json');
199        echo json_encode($response);
200    }
201
202    0 references | 0 overrides
203    public function changeEmail(){
204        $db = DB::getInstance();
205        $json = file_get_contents('php://input');
206
207        $request = json_decode($json, true);
208        $uid = $request['userID'];
209        $email = $request['uEmail'];
210
211        $query = "UPDATE users SET uEmail = ? WHERE userID = ?";
212        $stmt = $db->prepare($query);
213        $stmt->execute([$email, $uid]);
214
215        $response = ['Success'=> True];
216        header('Content-Type: application/json');
217        echo json_encode($response);
218    }
219
220    0 references | 0 overrides
221    public function changeBirthdate(){
222        $db = DB::getInstance();
223        $json = file_get_contents('php://input');
224
225        $request = json_decode($json, true);
226        $uid = $request['userID'];
227        $bday = $request['uBirthdate'];
228
229        $query = "UPDATE users SET uBirthdate = ? WHERE userID = ?";
230        $stmt = $db->prepare($query);
231        $stmt->execute([$bday, $uid]);
232
233        $response = ['Success'=> True];
234        header('Content-Type: application/json');
235        echo json_encode($response);
236    }
237
238    0 references | 0 overrides
239    public function changeImage(){
240        $db = DB::getInstance();
241        $json = file_get_contents('php://input');
242
243        $request = json_decode($json, true);
244        $uid = $request['userID'];
245        $image = $request['image'];
246
247        $query = "UPDATE users SET uProfilePicture = ? WHERE userID = ?";
248        $stmt = $db->prepare($query);
249        $stmt->execute([$image, $uid]);
250
251        $response = ['Success'=> True];
252        header('Content-Type: application/json');
253        echo json_encode($response);
254    }
255
256    0 references | 0 overrides
257    public function changeName(){
258        $db = DB::getInstance();
259        $json = file_get_contents('php://input');
260
261        $request = json_decode($json, true);
262        $uid = $request['userID'];
263        $name = $request['uName'];
264
265        $query = "UPDATE users SET uName = ? WHERE userID = ?";
266        $stmt = $db->prepare($query);
267        $stmt->execute([$name, $uid]);
268
269        $response = ['Success'=> True];
270        header('Content-Type: application/json');
271        echo json_encode($response);
272    }
273
274    0 references | 0 overrides
275    public function changeAddress(){
276        $db = DB::getInstance();
277        $json = file_get_contents('php://input');
278
279        $request = json_decode($json, true);
280        $uid = $request['userID'];
281        $address = $request['uAddress'];
282
283        $query = "UPDATE users SET uAddress = ? WHERE userID = ?";
284        $stmt = $db->prepare($query);
285        $stmt->execute([$address, $uid]);
286
287        $response = ['Success'=> True];
288        header('Content-Type: application/json');
289        echo json_encode($response);
290    }
291
292    0 references | 0 overrides
293    public function changePhone(){
294        $db = DB::getInstance();
295        $json = file_get_contents('php://input');
296
297        $request = json_decode($json, true);
298        $uid = $request['userID'];
299        $phone = $request['uPhone'];
300
301        $query = "UPDATE users SET uPhone = ? WHERE userID = ?";
302        $stmt = $db->prepare($query);
303        $stmt->execute([$phone, $uid]);
304
305        $response = ['Success'=> True];
306        header('Content-Type: application/json');
307        echo json_encode($response);
308    }
309
310    0 references | 0 overrides
311    public function changeEmail(){
312        $db = DB::getInstance();
313        $json = file_get_contents('php://input');
314
315        $request = json_decode($json, true);
316        $uid = $request['userID'];
317        $email = $request['uEmail'];
318
319        $query = "UPDATE users SET uEmail = ? WHERE userID = ?";
320        $stmt = $db->prepare($query);
321        $stmt->execute([$email, $uid]);
322
323        $response = ['Success'=> True];
324        header('Content-Type: application/json');
325        echo json_encode($response);
326    }
327
328    0 references | 0 overrides
329    public function changeBirthdate(){
330        $db = DB::getInstance();
331        $json = file_get_contents('php://input');
332
333        $request = json_decode($json, true);
334        $uid = $request['userID'];
335        $bday = $request['uBirthdate'];
336
337        $query = "UPDATE users SET uBirthdate = ? WHERE userID = ?";
338        $stmt = $db->prepare($query);
339        $stmt->execute([$bday, $uid]);
340
341        $response = ['Success'=> True];
342        header('Content-Type: application/json');
343        echo json_encode($response);
344    }
345
346    0 references | 0 overrides
347    public function changeImage(){
348        $db = DB::getInstance();
349        $json = file_get_contents('php://input');
350
351        $request = json_decode($json, true);
352        $uid = $request['userID'];
353        $image = $request['image'];
354
355        $query = "UPDATE users SET uProfilePicture = ? WHERE userID = ?";
356        $stmt = $db->prepare($query);
357        $stmt->execute([$image, $uid]);
358
359        $response = ['Success'=> True];
360        header('Content-Type: application/json');
361        echo json_encode($response);
362    }
363
364    0 references | 0 overrides
365    public function changeName(){
366        $db = DB::getInstance();
367        $json = file_get_contents('php://input');
368
369        $request = json_decode($json, true);
370        $uid = $request['userID'];
371        $name = $request['uName'];
372
373        $query = "UPDATE users SET uName = ? WHERE userID = ?";
374        $stmt = $db->prepare($query);
375        $stmt->execute([$name, $uid]);
376
377        $response = ['Success'=> True];
378        header('Content-Type: application/json');
379        echo json_encode($response);
380    }
381
382    0 references | 0 overrides
383    public function changeAddress(){
384        $db = DB::getInstance();
385        $json = file_get_contents('php://input');
386
387        $request = json_decode($json, true);
388        $uid = $request['userID'];
389        $address = $request['uAddress'];
390
391        $query = "UPDATE users SET uAddress = ? WHERE userID = ?";
392        $stmt = $db->prepare($query);
393        $stmt->execute([$address, $uid]);
394
395        $response = ['Success'=> True];
396        header('Content-Type: application/json');
397        echo json_encode($response);
398    }
399
400    0 references | 0 overrides
401    public function changePhone(){
402        $db = DB::getInstance();
403        $json = file_get_contents('php://input');
404
405        $request = json_decode($json, true);
406        $uid = $request['userID'];
407        $phone = $request['uPhone'];
408
409        $query = "UPDATE users SET uPhone = ? WHERE userID = ?";
410        $stmt = $db->prepare($query);
411        $stmt->execute([$phone, $uid]);
412
413        $response = ['Success'=> True];
414        header('Content-Type: application/json');
415        echo json_encode($response);
416    }
417
418    0 references | 0 overrides
419    public function changeEmail(){
420        $db = DB::getInstance();
421        $json = file_get_contents('php://input');
422
423        $request = json_decode($json, true);
424        $uid = $request['userID'];
425        $email = $request['uEmail'];
426
427        $query = "UPDATE users SET uEmail = ? WHERE userID = ?";
428        $stmt = $db->prepare($query);
429        $stmt->execute([$email, $uid]);
430
431        $response = ['Success'=> True];
432        header('Content-Type: application/json');
433        echo json_encode($response);
434    }
435
436    0 references | 0 overrides
437    public function changeBirthdate(){
438        $db = DB::getInstance();
439        $json = file_get_contents('php://input');
440
441        $request = json_decode($json, true);
442        $uid = $request['userID'];
443        $bday = $request['uBirthdate'];
444
445        $query = "UPDATE users SET uBirthdate = ? WHERE userID = ?";
446        $stmt = $db->prepare($query);
447        $stmt->execute([$bday, $uid]);
448
449        $response = ['Success'=> True];
450        header('Content-Type: application/json');
451        echo json_encode($response);
452    }
453
454    0 references | 0 overrides
455    public function changeImage(){
456        $db = DB::getInstance();
457        $json = file_get_contents('php://input');
458
459        $request = json_decode($json, true);
460        $uid = $request['userID'];
461        $image = $request['image'];
462
463        $query = "UPDATE users SET uProfilePicture = ? WHERE userID = ?";
464        $stmt = $db->prepare($query);
465        $stmt->execute([$image, $uid]);
466
467        $response = ['Success'=> True];
468        header('Content-Type: application/json');
469        echo json_encode($response);
470    }
471
472    0 references | 0 overrides
473    public function changeName(){
474        $db = DB::getInstance();
475        $json = file_get_contents('php://input');
476
477        $request = json_decode($json, true);
478        $uid = $request['userID'];
479        $name = $request['uName'];
480
481        $query = "UPDATE users SET uName = ? WHERE userID = ?";
482        $stmt = $db->prepare($query);
483        $stmt->execute([$name, $uid]);
484
485        $response = ['Success'=> True];
486        header('Content-Type: application/json');
487        echo json_encode($response);
488    }
489
490    0 references | 0 overrides
491    public function changeAddress(){
492        $db = DB::getInstance();
493        $json = file_get_contents('php://input');
494
495        $request = json_decode($json, true);
496        $uid = $request['userID'];
497        $address = $request['uAddress'];
498
499        $query = "UPDATE users SET uAddress = ? WHERE userID = ?";
500        $stmt = $db->prepare($query);
501        $stmt->execute([$address, $uid]);
502
503        $response = ['Success'=> True];
504        header('Content-Type: application/json');
505        echo json_encode($response);
506    }
507
508    0 references | 0 overrides
509    public function changePhone(){
510        $db = DB::getInstance();
511        $json = file_get_contents('php://input');
512
513        $request = json_decode($json, true);
514        $uid = $request['userID'];
515        $phone = $request['uPhone'];
516
517        $query = "UPDATE users SET uPhone = ? WHERE userID = ?";
518        $stmt = $db->prepare($query);
519        $stmt->execute([$phone, $uid]);
520
521        $response = ['Success'=> True];
522        header('Content-Type: application/json');
523        echo json_encode($response);
524    }
525
526    0 references | 0 overrides
527    public function changeEmail(){
528        $db = DB::getInstance();
529        $json = file_get_contents('php://input');
530
531        $request = json_decode($json, true);
532        $uid = $request['userID'];
533        $email = $request['uEmail'];
534
535        $query = "UPDATE users SET uEmail = ? WHERE userID = ?";
536        $stmt = $db->prepare($query);
537        $stmt->execute([$email, $uid]);
538
539        $response = ['Success'=> True];
540        header('Content-Type: application/json');
541        echo json_encode($response);
542    }
543
544    0 references | 0 overrides
545    public function changeBirthdate(){
546        $db = DB::getInstance();
547        $json = file_get_contents('php://input');
548
549        $request = json_decode($json, true);
550        $uid = $request['userID'];
551        $bday = $request['uBirthdate'];
552
553        $query = "UPDATE users SET uBirthdate = ? WHERE userID = ?";
554        $stmt = $db->prepare($query);
555        $stmt->execute([$bday, $uid]);
556
557        $response = ['Success'=> True];
558        header('Content-Type: application/json');
559        echo json_encode($response);
560    }
561
562    0 references | 0 overrides
563    public function changeImage(){
564        $db = DB::getInstance();
565        $json = file_get_contents('php://input');
566
567        $request = json_decode($json, true);
568        $uid = $request['userID'];
569        $image = $request['image'];
570
571        $query = "UPDATE users SET uProfilePicture = ? WHERE userID = ?";
572        $stmt = $db->prepare($query);
573        $stmt->execute([$image, $uid]);
574
575        $response = ['Success'=> True];
576        header('Content-Type: application/json');
577        echo json_encode($response);
578    }
579
580    0 references | 0 overrides
581    public function changeName(){
582        $db = DB::getInstance();
583        $json = file_get_contents('php://input');
584
585        $request = json_decode($json, true);
586        $uid = $request['userID'];
587        $name = $request['uName'];
588
589        $query = "UPDATE users SET uName = ? WHERE userID = ?";
590        $stmt = $db->prepare($query);
591        $stmt->execute([$name, $uid]);
592
593        $response = ['Success'=> True];
594        header('Content-Type: application/json');
595        echo json_encode($response);
596    }
597
598    0 references | 0 overrides
599    public function changeAddress(){
600        $db = DB::getInstance();
601        $json = file_get_contents('php://input');
602
603        $request = json_decode($json, true);
604        $uid = $request['userID'];
605        $address = $request['uAddress'];
606
607        $query = "UPDATE users SET uAddress = ? WHERE userID = ?";
608        $stmt = $db->prepare($query);
609        $stmt->execute([$address, $uid]);
610
611        $response = ['Success'=> True];
612        header('Content-Type: application/json');
613        echo json_encode($response);
614    }
615
616    0 references | 0 overrides
617    public function changePhone(){
618        $db = DB::getInstance();
619        $json = file_get_contents('php://input');
620
621        $request = json_decode($json, true);
622        $uid = $request['userID'];
623        $phone = $request['uPhone'];
624
625        $query = "UPDATE users SET uPhone = ? WHERE userID = ?";
626        $stmt = $db->prepare($query);
627        $stmt->execute([$phone, $uid]);
628
629        $response = ['Success'=> True];
630        header('Content-Type: application/json');
631        echo json_encode($response);
632    }
633
634    0 references | 0 overrides
635    public function changeEmail(){
636        $db = DB::getInstance();
637        $json = file_get_contents('php://input');
638
639        $request = json_decode($json, true);
640        $uid = $request['userID'];
641        $email = $request['uEmail'];
642
643        $query = "UPDATE users SET uEmail = ? WHERE userID = ?";
644        $stmt = $db->prepare($query);
645        $stmt->execute([$email, $uid]);
646
647        $response = ['Success'=> True];
648        header('Content-Type: application/json');
649        echo json_encode($response);
650    }
651
652    0 references | 0 overrides
653    public function changeBirthdate(){
654        $db = DB::getInstance();
655        $json = file_get_contents('php://input');
656
657        $request = json_decode($json, true);
658        $uid = $request['userID'];
659        $bday = $request['uBirthdate'];
660
661        $query = "UPDATE users SET uBirthdate = ? WHERE userID = ?";
662        $stmt = $db->prepare($query);
663        $stmt->execute([$bday, $uid]);
664
665        $response = ['Success'=> True];
666        header('Content-Type: application/json');
667        echo json_encode($response);
668    }
669
670    0 references | 0 overrides
671    public function changeImage(){
672        $db = DB::getInstance();
673        $json = file_get_contents('php://input');
674
675        $request = json_decode($json, true);
676        $uid = $request['userID'];
677        $image = $request['image'];
678
679        $query = "UPDATE users SET uProfilePicture = ? WHERE userID = ?";
680        $stmt = $db->prepare($query);
681        $stmt->execute([$image, $uid]);
682
683        $response = ['Success'=> True];
684        header('Content-Type: application/json');
685        echo json_encode($response);
686    }
687
688    0 references | 0 overrides
689    public function changeName(){
690        $db = DB::getInstance();
691        $json = file_get_contents('php://input');
692
693        $request = json_decode($json, true);
694        $uid = $request['userID'];
695        $name = $request['uName'];
696
697        $query = "UPDATE users SET uName = ? WHERE userID = ?";
698        $stmt = $db->prepare($query);
699        $stmt->execute([$name, $uid]);
700
701        $response = ['Success'=> True];
702        header('Content-Type: application/json');
703        echo json_encode($response);
704    }
705
706    0 references | 0 overrides
707    public function changeAddress(){
708        $db = DB::getInstance();
709        $json = file_get_contents('php://input');
710
711        $request = json_decode($json, true);
712        $uid = $request['userID'];
713        $address = $request['uAddress'];
714
715        $query = "UPDATE users SET uAddress = ? WHERE userID = ?";
716        $stmt = $db->prepare($query);
717        $stmt->execute([$address, $uid]);
718
719        $response = ['Success'=> True];
720        header('Content-Type: application/json');
721        echo json_encode($response);
722    }
723
724    0 references | 0 overrides
725    public function changePhone(){
726        $db = DB::getInstance();
727        $json = file_get_contents('php://input');
728
729        $request = json_decode($json, true);
730        $uid = $request['userID'];
731        $phone = $request['uPhone'];
732
733        $query = "UPDATE users SET uPhone = ? WHERE userID = ?";
734        $stmt = $db->prepare($query);
735        $stmt->execute([$phone, $uid]);
736
737        $response = ['Success'=> True];
738        header('Content-Type: application/json');
739        echo json_encode($response);
740    }
741
742    0 references | 0 overrides
743    public function changeEmail(){
744        $db = DB::getInstance();
745        $json = file_get_contents('php://input');
746
747        $request = json_decode($json, true);
748        $uid = $request['userID'];
749        $email = $request['uEmail'];
750
751        $query = "UPDATE users SET uEmail = ? WHERE userID = ?";
752        $stmt = $db->prepare($query);
753        $stmt->execute([$email, $uid]);
754
755        $response = ['Success'=> True];
756        header('Content-Type: application/json');
757        echo json_encode($response);
758    }
759
760    0 references | 0 overrides
761    public function changeBirthdate(){
762        $db = DB::getInstance();
763        $json = file_get_contents('php://input');
764
765        $request = json_decode($json, true);
766        $uid = $request['userID'];
767        $bday = $request['uBirthdate'];
768
769        $query = "UPDATE users SET uBirthdate = ? WHERE userID = ?";
770        $stmt = $db->prepare($query);
771        $stmt->execute([$bday, $uid]);
772
773        $response = ['Success'=> True];
774        header('Content-Type: application/json');
775        echo json_encode($response);
776    }
777
778    0 references | 0 overrides
779    public function changeImage(){
780        $db = DB::getInstance();
781        $json = file_get_contents('php://input');
782
783        $request = json_decode($json, true);
784        $uid = $request['userID'];
785        $image = $request['image'];
786
787        $query = "UPDATE users SET uProfilePicture = ? WHERE userID = ?";
788        $stmt = $db->prepare($query);
789        $stmt->execute([$image, $uid]);
790
791        $response = ['Success'=> True];
792        header('Content-Type: application/json');
793        echo json_encode($response);
794    }
795
796    0 references | 0 overrides
797    public function changeName(){
798        $db = DB::getInstance();
799        $json = file_get_contents('php://input');
800
801        $request = json_decode($json, true);
802        $uid = $request['userID'];
803        $name = $request['uName'];
804
805        $query = "UPDATE users SET uName = ? WHERE userID = ?";
806        $stmt = $db->prepare($query);
807        $stmt->execute([$name, $uid]);
808
809        $response = ['Success'=> True];
810        header('Content-Type: application/json');
811        echo json_encode($response);
812    }
813
814    0 references | 0 overrides
815    public function changeAddress(){
816        $db = DB::getInstance();
817        $json = file_get_contents('php://input');
818
819        $request = json_decode($json, true);
820        $uid = $request['userID'];
821        $address = $request['uAddress'];
822
823        $query = "UPDATE users SET uAddress = ? WHERE userID = ?";
824        $stmt = $db->prepare($query);
825        $stmt->execute([$address, $uid]);
826
827        $response = ['Success'=> True];
828        header('Content-Type: application/json');
829        echo json_encode($response);
830    }
831
832    0 references | 0 overrides
833    public function changePhone(){
834        $db = DB::getInstance();
835        $json = file_get_contents('php://input');
836
837        $request = json_decode($json, true);
838        $uid = $request['userID'];
839        $phone = $request['uPhone'];
840
841        $query = "UPDATE users SET uPhone = ? WHERE userID = ?";
842        $stmt = $db->prepare($query);
843        $stmt->execute([$phone, $uid]);
844
845        $response = ['Success'=> True];
846        header('Content-Type: application/json');
847        echo json_encode($response);
848    }
849
850    0 references | 0 overrides
851    public function changeEmail(){
852        $db = DB::getInstance();
853        $json = file_get_contents('php://input');
854
855        $request = json_decode($json, true);
856        $uid = $request['userID'];
857        $email = $request['uEmail'];
858
859        $query = "UPDATE users SET uEmail = ? WHERE userID = ?";
860        $stmt = $db->prepare($query);
861        $stmt->execute([$email, $uid]);
862
863        $response = ['Success'=> True];
864        header('Content-Type: application/json');
865        echo json_encode($response);
866    }
867
868    0 references | 0 overrides
869    public function changeBirthdate(){
870        $db = DB::getInstance();
871        $json = file_get_contents('php://input');
872
873        $request = json_decode($json, true);
874        $uid = $request['userID'];
875        $bday = $request['uBirthdate'];
876
877        $query = "UPDATE users SET uBirthdate = ? WHERE userID = ?";
878        $stmt = $db->prepare($query);
879        $stmt->execute([$bday, $uid]);
880
881        $response = ['Success'=> True];
882        header('Content-Type: application/json');
883        echo json_encode($response);
884    }
885
886    0 references | 0 overrides
887    public function changeImage(){
888        $db = DB::getInstance();
889        $json = file_get_contents('php://input');
890
891        $request = json_decode($json, true);
892        $uid = $request['userID'];
893        $image = $request['image'];
894
895        $query = "UPDATE users SET uProfilePicture = ? WHERE userID = ?";
896        $stmt = $db->prepare($query);
897        $stmt->execute([$image, $uid]);
898
899        $response = ['Success'=> True];
900        header('Content-Type: application/json');
901        echo json_encode($response);
902    }
903
904    0 references | 0 overrides
905    public function changeName(){
906        $db = DB::getInstance();
907        $json = file_get_contents('php://input');
908
909        $request = json_decode($json, true);
910        $uid = $request['userID'];
911        $name = $request['uName'];
912
913        $query = "UPDATE users SET uName = ? WHERE userID = ?";
914        $stmt = $db->prepare($query);
915        $stmt->execute([$name, $uid]);
916
917        $response = ['Success'=> True];
918        header('Content-Type: application/json');
919        echo json_encode($response);
920    }
921
922    0 references | 0 overrides
923    public function changeAddress(){
924        $db = DB::getInstance();
925        $json = file_get_contents('php://input');
926
927        $request = json_decode($json, true);
928        $uid = $request['userID'];
929        $address = $request['uAddress'];
930
931        $query = "UPDATE users SET uAddress = ? WHERE userID = ?";
932        $stmt = $db->prepare($query);
933        $stmt->execute([$address, $uid]);
934
935        $response = ['Success'=> True];
936        header('Content-Type: application/json');
937        echo json_encode($response);
938    }
939
940    0 references | 0 overrides
941    public function changePhone(){
942        $db = DB::getInstance();
943        $json = file_get_contents('php://input');
944
945        $request = json_decode($json, true);
946        $uid = $request['userID'];
947        $phone = $request['uPhone'];
948
949        $query = "UPDATE users SET uPhone = ? WHERE userID = ?";
950        $stmt = $db->prepare($query);
951        $stmt->execute([$phone, $uid]);
952
953        $response = ['Success'=> True];
954        header('Content-Type: application/json');
955        echo json_encode($response);
956    }
957
958    0 references | 0 overrides
959    public function changeEmail(){
960        $db = DB::getInstance();
961        $json = file_get_contents('php://input');
962
963        $request = json_decode($json, true);
964        $uid = $request['userID'];
965        $email = $request['uEmail'];
966
967        $query = "UPDATE users SET uEmail = ? WHERE userID = ?";
968        $stmt = $db->prepare($query);
969        $stmt->execute([$email, $uid]);
970
971        $response = ['Success'=> True];
972        header('Content-Type: application/json');
973        echo json_encode($response);
974    }
975
976    0 references | 0 overrides
977    public function changeBirthdate(){
978        $db = DB::getInstance();
979        $json = file_get_contents('php://input');
980
981        $request = json_decode($json, true);
982        $uid = $request['userID'];
983        $bday = $request['uBirthdate'];
984
985        $query = "UPDATE users SET uBirthdate = ? WHERE userID = ?";
986        $stmt = $db->prepare($query);
987        $stmt->execute([$bday, $uid]);
988
989        $response = ['Success'=> True];
990        header('Content-Type: application/json');
991        echo json_encode($response);
992    }
993
994    0 references | 0 overrides
995    public function changeImage(){
996        $db = DB::getInstance();
997        $json = file_get_contents('php://input');
998
999        $request = json_decode($json, true);
1000       $uid = $request['userID'];
1001       $image = $request['image'];
1002
1003       $query = "UPDATE users SET uProfilePicture = ? WHERE userID = ?";
1004       $stmt = $db->prepare($query);
1005       $stmt->execute([$image, $uid]);
1006
1007       $response = ['Success'=> True];
1008       header('Content-Type: application/json');
1009       echo json_encode($response);
1010   }
1011
1012   0 references | 0 overrides
1013   public function changeName(){
1014       $db = DB::getInstance();
1015       $json = file_get_contents('php://input');
1016
1017       $request = json_decode($json, true);
1018       $uid = $request['userID'];
1019       $name = $request['uName'];
1020
1021       $query = "UPDATE users SET uName = ? WHERE userID = ?";
1022       $stmt = $db->prepare($query);
1023       $stmt->execute([$name, $uid]);
1024
1025       $response = ['Success'=> True];
1026       header('Content-Type: application/json');
1027       echo json_encode($response);
1028   }
1029
1030   0 references | 0 overrides
1031   public function changeAddress(){
1032       $db = DB::getInstance();
1033       $json = file_get_contents('php://input');
1034
1035       $request = json_decode($json, true);
1036       $uid = $request['userID'];
1037       $address = $request['uAddress'];
1038
1039       $query = "UPDATE users SET uAddress = ? WHERE userID = ?";
1040       $stmt = $db->prepare($query);
1041       $stmt->execute([$address, $uid]);
1042
1043       $response = ['Success'=> True];
1044       header('Content-Type: application/json');
1045       echo json_encode($response);
1046   }
1047
1048   0 references | 0 overrides
1049   public function changePhone(){
1050       $db = DB::getInstance();
1051       $json = file_get_contents('php://input');
1052
1053       $request = json_decode($json, true);
1054       $uid = $request['userID'];
1055       $phone = $request['uPhone'];
1056
1057       $query = "UPDATE users SET uPhone = ? WHERE userID = ?";
1058       $stmt = $db->prepare($query);
1059       $stmt->execute([$phone, $uid]);
1060
1061       $response = ['Success'=> True];
1062       header('Content-Type: application/json');
1063       echo json_encode($response);
1064   }
1065
1066   0 references | 0 overrides
1067   public function changeEmail(){
1068       $db = DB::getInstance();
1069       $json = file_get_contents('php://input');
1070
1071       $request = json_decode($json, true);
1072       $uid = $request['userID'];
1073       $email = $request['uEmail'];
1074
1075       $query = "UPDATE users SET uEmail = ? WHERE userID = ?";
1076       $stmt = $db->prepare($query);
1077       $stmt->execute([$email, $uid]);
1078
1079       $response = ['Success'=> True];
1080       header('Content-Type: application/json');
1081       echo json_encode($response);
1082   }
1083
1084   0 references | 0 overrides
1085   public function changeBirthdate(){
1086       $db = DB::getInstance();
1087       $json = file_get_contents('php://input');
1088
1089       $request = json_decode($json, true);
1090       $uid = $request['userID'];
1091       $bday = $request['uBirthdate'];
1092
1093       $query = "UPDATE users SET uBirthdate = ? WHERE userID = ?";
1094       $stmt = $db->prepare($query);
1095       $stmt->execute([$bday, $uid]);
1096
1097       $response = ['Success'=> True];
1098       header('Content-Type: application/json');
1099       echo json_encode($response);
1100   }
1101
1102   0 references | 0 overrides
1103   public function changeImage(){
1104       $db = DB::getInstance();
1105       $json = file_get_contents('php://input');
1106
1107       $request = json_decode($json, true);
1108       $uid = $request['userID'];
1109       $image = $request['image'];
1110
1111       $query = "UPDATE users SET uProfilePicture = ? WHERE userID = ?";
1112       $stmt = $db->prepare($query);
1113       $stmt->execute([$image, $uid]);
1114
1115       $response = ['Success'=> True];
1116       header('Content-Type: application/json');
1117       echo json_encode($response);
1118   }
1119
1120   0 references | 0 overrides
1121   public function changeName(){
1122       $db = DB::getInstance();
1123       $json = file_get_contents('php://
```

```

	accountController.php X
C: > xampp > htdocs > ezgo > accountController.php > ...
4   class account{
30     public function updateAccount(){
43       $file_name = basename($data['image']['name']);
44       $file_path = 'images/' . $file_name;
45
46       if (file_put_contents($file_path, base64_decode($data['image']['data']))) {
47         $fields = array('uName', 'uAddress', 'uPhone', 'uEmail', 'uBirthdate', 'uProfilePicture');
48         $values = array($name, $address, $phone, $email, $bday, $file_name);
49         $assoc_array = array_combine($fields, $values);
50
51         $where = array('userID', $uid);
52
53         if($db->update('users', $assoc_array, $where)){
54           $response = ['Success'=> True];
55           header('Content-Type: application/json');
56           echo json_encode($response);
57         }else{
58           $response = ['Success'=> False];
59           header('Content-Type: application/json');
60           echo json_encode($response);
61         }
62       }else{
63         $response = ['Success'=> False];
64         header('Content-Type: application/json');
65         echo json_encode($response);
66       }
67     }
68
69     0 references | 0 overrides
70     public function delete(){
71       $db = DB::getInstance();
72       $json = file_get_contents('php://input');
73
74       $request = json_decode($json, true);
75       $uid = $request['userID'];
76
77       $where = array('userID', '=', $uid);
78       if($db->delete('users', $where)){
79         $response = ['Success'=> True];
80         header('Content-Type: application/json');
81         echo json_encode($response);
82       }else{
83         $response = ['Success'=> False];
84         header('Content-Type: application/json');
85         echo json_encode($response);
86       }
87     }
88   ?>

```

Kode PHP accountController di atas merupakan bagian dari controller yang mengelola akun pengguna dalam suatu aplikasi. Controller ini menyediakan tiga fungsi utama: *showAccount()*, *updateAccount()*, dan *delete()*. Fungsi *showAccount()* digunakan untuk menampilkan informasi akun pengguna berdasarkan *userID* yang diterima melalui permintaan

HTTP, termasuk menggabungkan jalur gambar profil pengguna. Fungsi *updateAccount()* memperbarui informasi akun pengguna, termasuk menyimpan gambar profil baru yang dikirimkan dalam bentuk data base64 dan memperbarui informasi lain seperti nama, alamat, telepon, email, dan tanggal lahir. Fungsi *delete()* menghapus akun pengguna berdasarkan *userID* yang diterima. Semua fungsi tersebut berinteraksi dengan database menggunakan kelas *DB* untuk melakukan operasi CRUD (Create, Read, Update, Delete). Kode ini mengimplementasikan fitur manajemen akun pengguna untuk aplikasi yang memerlukan autentikasi dan profil pengguna.

### ***3.3.2. cityController.php***

```
cityController.php x
C: > xampp > htdocs > ezgo > cityController.php > PHP > city > getCityTwo()
1  <?php
2  include_once 'database.php';
3
4  class city{
5      0 references | 0 implementations
6      0 references | 0 overrides
7      public function list(){
8          $db = DB::getInstance();
9
10         $city = $db->select('*', 'city')->getResult();
11
12         if(count($city) > 0){
13             $response = ['Success'=> True, 'Data' => $city];
14             header('Content-Type: application/json');
15             echo json_encode($response);
16         }else{
17             $response = ['Success'=> False];
18             header('Content-Type: application/json');
19             echo json_encode($response);
20         }
21
22         0 references | 0 overrides
23         public function getCity(){
24             $db = DB::getInstance();
25             $json = file_get_contents('php://input');
26
27             $request = json_decode($json, true);
28             $cid = $request['cityID'];
29
30             $where = array('cityID', '=', $cid);
31             $cityName = $db->select('cName', 'city', $where)->getResult();
32
33             if(count($cityName) > 0){
34                 $response = ['Success'=> True, 'Data' => $cityName[0]->cName];
35                 header('Content-Type: application/json');
36                 echo json_encode($response);
37             }else{
38                 $response = ['Success'=> False];
39                 header('Content-Type: application/json');
40                 echo json_encode($response);
41             }
42
43             0 references | 0 overrides
44             public function getCityTwo()[]
```

```

cityController.php ×
C: > xampp > htdocs > ezgo > cityController.php > ...
4   class city{
42     public function getCityTwo(){
43       $db = DB::getInstance();
44       $json = file_get_contents('php://input');
45
46       $request = json_decode($json, true);
47       $cid1 = $request['cityIDfrom'];
48       $cid2 = $request['cityIDdest'];
49
50       $where = array('cityID', '=', $cid1);
51       $cityName1 = $db->select('cName', 'city', $where)->getResult();
52
53       $where = array('cityID', '=', $cid2);
54       $cityName2 = $db->select('cName', 'city', $where)->getResult();
55
56       if(count($cityName1) && count($cityName2)){
57         $names = array($cityName1[0]->cName, $cityName2[0]->cName);
58         $response = ['Success'=> True, 'Data' => $names];
59         header('Content-Type: application/json');
60         echo json_encode($response);
61       }else{
62         $response = ['Success'=> False];
63         header('Content-Type: application/json');
64         echo json_encode($response);
65       }
66     }
67   }
68 ?>

```

Kode PHP cityController di atas merupakan bagian dari controller yang mengelola data kota dalam suatu aplikasi. Controller ini menyediakan tiga fungsi utama: *list()*, *getCity()*, dan *getCityTwo()*. Fungsi *list()* digunakan untuk mengambil dan menampilkan semua data kota dari tabel *city*. Fungsi *getCity()* mengambil dan menampilkan nama kota berdasarkan *cityID* yang diterima dari permintaan HTTP. Fungsi *getCityTwo()* mengambil dan menampilkan nama dua kota berdasarkan *cityIDfrom* dan *cityIDdest* yang diterima dari permintaan HTTP. Semua fungsi ini berinteraksi dengan database menggunakan kelas *DB* untuk melakukan operasi pengambilan data. Kode ini mengimplementasikan fitur manajemen data kota, yang esensial untuk aplikasi yang memerlukan informasi kota seperti pada layanan perjalanan atau peta.

### 3.3.3. database.php

```
database.php X

C: > xampp > htdocs > ezgo > database.php > PHP > DB > isRectangular()

1  <?php
2      10 references | 0 implementations
3  class DB{
4      3 references
5      private static $_instance = null;
6      private $_pdo, $_query, $_error = false, $_results, $_count = 0;
7
8      1 reference
9      private function __construct(){
10          try{
11              $this->_pdo = new PDO('mysql:host=localhost;dbname=id21634510_ezgo;', 'root', '');
12          }catch(PDOException $e){
13              die($e->getMessage());
14          }
15      }
16
17      6 references | 0 overrides
18      public static function getInstance(){
19          if(!isset(self::$_instance)){
20              self::$_instance = new DB();
21          }
22          return self::$_instance;
23      }
24
25      4 references | 0 overrides
26      public function query($sql, $params = array()){
27          $this->_error = false;
28          if($this->_query = $this->_pdo->prepare($sql)){
29              $x=1;
30              if(count($params)){
31                  foreach($params as $param){
32                      $this->_query->bindValue($x, $param);
33                      $x++;
34                  }
35              }
36
37              if($this->_query->execute()){
38                  $this->_results = $this->_query->fetchAll(PDO::FETCH_OBJ);
39                  $this->_count = $this->_query->rowCount();
40              }else{
41                  $this->_error = true;
42              }
43              return $this;
44          }
45
46          0 references | 0 overrides
47          public function getResult(){
48              return $this->_results;
49          }
50
51          4 references | 0 overrides
52          public function error(){
53              return $this->_error;
54          }
55
56          0 references | 0 overrides
57          public function __destruct(){
58              $this->close();
59          }
60
61          0 references | 0 overrides
62          public function __clone(){
63              $this->close();
64          }
65
66          0 references | 0 overrides
67          public function __wakeup(){
68              $this->close();
69          }
70
71          0 references | 0 overrides
72          public function __sleep(){
73              $this->close();
74          }
75
76          0 references | 0 overrides
77          public function __toString(){
78              return $this->error();
79          }
80
81          0 references | 0 overrides
82          public function __get($name){
83              if($name == '_error'){
84                  return $this->_error;
85              }
86              if($name == '_results'){
87                  return $this->_results;
88              }
89              if($name == '_count'){
90                  return $this->_count;
91              }
92              if($name == '_query'){
93                  return $this->_query;
94              }
95              if($name == '_pdo'){
96                  return $this->_pdo;
97              }
98
99              $this->error("Unknown property: $name");
100         }
101
102         0 references | 0 overrides
103         public function __set($name, $value){
104             if($name == '_error'){
105                 $this->_error = $value;
106             }
107             if($name == '_results'){
108                 $this->_results = $value;
109             }
110             if($name == '_count'){
111                 $this->_count = $value;
112             }
113             if($name == '_query'){
114                 $this->_query = $value;
115             }
116             if($name == '_pdo'){
117                 $this->_pdo = $value;
118             }
119
120             $this->error("Unknown property: $name");
121         }
122
123         0 references | 0 overrides
124         public function __isset($name){
125             if($name == '_error'){
126                 return $this->_error;
127             }
128             if($name == '_results'){
129                 return $this->_results;
130             }
131             if($name == '_count'){
132                 return $this->_count;
133             }
134             if($name == '_query'){
135                 return $this->_query;
136             }
137             if($name == '_pdo'){
138                 return $this->_pdo;
139             }
140
141             $this->error("Unknown property: $name");
142         }
143
144         0 references | 0 overrides
145         public function __unset($name){
146             if($name == '_error'){
147                 $this->_error = null;
148             }
149             if($name == '_results'){
150                 $this->_results = null;
151             }
152             if($name == '_count'){
153                 $this->_count = null;
154             }
155             if($name == '_query'){
156                 $this->_query = null;
157             }
158             if($name == '_pdo'){
159                 $this->_pdo = null;
160             }
161
162             $this->error("Unknown property: $name");
163         }
164
165         0 references | 0 overrides
166         public function __call($name, $arguments){
167             $this->error("Unknown method: $name");
168         }
169
170         0 references | 0 overrides
171         public function __wakeup(){
172             $this->error("Unknown method: __wakeup");
173         }
174
175         0 references | 0 overrides
176         public function __sleep(){
177             $this->error("Unknown method: __sleep");
178         }
179
180         0 references | 0 overrides
181         public function __clone(){
182             $this->error("Unknown method: __clone");
183         }
184
185         0 references | 0 overrides
186         public function __destruct(){
187             $this->error("Unknown method: __destruct");
188         }
189
190         0 references | 0 overrides
191         public function __toString(){
192             $this->error("Unknown method: __toString");
193         }
194
195         0 references | 0 overrides
196         public function __get($name){
197             $this->error("Unknown method: __get");
198         }
199
200         0 references | 0 overrides
201         public function __set($name, $value){
202             $this->error("Unknown method: __set");
203         }
204
205         0 references | 0 overrides
206         public function __isset($name){
207             $this->error("Unknown method: __isset");
208         }
209
210         0 references | 0 overrides
211         public function __unset($name){
212             $this->error("Unknown method: __unset");
213         }
214
215         0 references | 0 overrides
216         public function __call($name, $arguments){
217             $this->error("Unknown method: __call");
218         }
219
220         0 references | 0 overrides
221         public function __wakeup(){
222             $this->error("Unknown method: __wakeup");
223         }
224
225         0 references | 0 overrides
226         public function __sleep(){
227             $this->error("Unknown method: __sleep");
228         }
229
230         0 references | 0 overrides
231         public function __clone(){
232             $this->error("Unknown method: __clone");
233         }
234
235         0 references | 0 overrides
236         public function __destruct(){
237             $this->error("Unknown method: __destruct");
238         }
239
240         0 references | 0 overrides
241         public function __toString(){
242             $this->error("Unknown method: __toString");
243         }
244
245         0 references | 0 overrides
246         public function __get($name){
247             $this->error("Unknown method: __get");
248         }
249
250         0 references | 0 overrides
251         public function __set($name, $value){
252             $this->error("Unknown method: __set");
253         }
254
255         0 references | 0 overrides
256         public function __isset($name){
257             $this->error("Unknown method: __isset");
258         }
259
260         0 references | 0 overrides
261         public function __unset($name){
262             $this->error("Unknown method: __unset");
263         }
264
265         0 references | 0 overrides
266         public function __call($name, $arguments){
267             $this->error("Unknown method: __call");
268         }
269
270         0 references | 0 overrides
271         public function __wakeup(){
272             $this->error("Unknown method: __wakeup");
273         }
274
275         0 references | 0 overrides
276         public function __sleep(){
277             $this->error("Unknown method: __sleep");
278         }
279
280         0 references | 0 overrides
281         public function __clone(){
282             $this->error("Unknown method: __clone");
283         }
284
285         0 references | 0 overrides
286         public function __destruct(){
287             $this->error("Unknown method: __destruct");
288         }
289
290         0 references | 0 overrides
291         public function __toString(){
292             $this->error("Unknown method: __toString");
293         }
294
295         0 references | 0 overrides
296         public function __get($name){
297             $this->error("Unknown method: __get");
298         }
299
299 }
```

database.php X

C: > xampp > htdocs > ezgo > database.php > PHP > DB > select()

```
2 class DB{
49
50     3 references | 0 overrides
51     public function isRectangular($array = array()) {
52         if (!is_array($array) || empty($array)) {
53             return false;
54         }
55         if (!is_array(reset($array))) {
56             return false;
57         }
58         try {
59             $x = count(array_unique(array_map(function($elem) {
60                 return is_array($elem) || $elem instanceof Countable ? count($elem) : 0;
61             }, $array)));
62             if ($x === 1) {
63                 return true;
64             }
65         } catch (Exception | Error $e) {
66             return false;
67         } catch (Throwable $t) {
68             return false;
69         } catch (TypeError $err) {
70             return false;
71         }
72     }
73
74     1 reference | 0 overrides
75     public function select($getField, $table, $where = null, $limit = null, $orderby = null, $orderDirection = 'ASC') {
76         $sql = "SELECT {$getField} FROM {$table}";
77
78         $value = array();
79
80         if (is_array($where)) {
81             if (count($where) === 3 && !$this->isRectangular($where)) {
82                 $operators = array('=', '>', '<', '>=', '<=');
83
84                 $field = $where[0];
85                 $operator = $where[1];
86                 $value = array($where[2]);
87
88                 if (in_array($operator, $operators)) {
89                     $sql .= " WHERE {$field} {$operator} ?";
90                 }
91             } else if ($this->isRectangular($where)) {
92                 $sql .= " WHERE ";
93                 $conditions = array();
94                 foreach ($where[0] as $index => $field) {
95                     $operator = $where[1][$index];
96                     $val = $where[2][$index];
97
98                     $conditions[] = "{$field} {$operator} ?";
99                     $value[] = $val;
100                }
101            }
102        }
103    }
104}
```

database.php X

C: > xampp > htdocs > ezgo > database.php > PHP > DB > update()

```
2     class DB{
74         public function select($getField, $table, $where = null, $limit = null, $orderby = null, $orderDirection = 'ASC') {
104            if (!is_null($orderby)) {
105                $sql .= " ORDER BY {$orderby} {$orderDirection}";
106            }
107
108            if (!is_null($limit)) {
109                $sql .= " LIMIT {$limit}";
110            }
111
112            if (!$this->query($sql, $value)->error()) {
113                return $this;
114            }
115
116            return false;
117        }
118
119
120        0 references | 0 overrides
121        public function insert($table, $fields = array()){
122            if(count($fields)){
123                $keys = array_keys($fields);
124                $values = null;
125                $x = 1;
126
127                foreach($fields as $field){
128                    $values .= "?";
129                    if($x < count($fields)){
130                        $values .= ", ";
131                    }
132                    $x++;
133                }
134
135                $sql = "INSERT INTO {$table} (`".implode('` , `', $keys)."`) VALUES (".$values.")";
136
137                if(!$this->query($sql, $fields)->error()){
138                    return true;
139                }
140            }
141            return false;
142        }
143
144        1 reference | 0 overrides
145        public function update($table, $fields, $where){
146            if(count($fields) && count($where)){
147                $set = null;
148                $keys = array_keys($fields);
149                $whereKeys = array_keys($where);
150                $x = 1;
151
152                foreach($keys as $key){
153                    $set .= "{$key} = ?";
154                    if($x < count($fields)){
155                        $set .= ", ";
156                    }
157                    $x++;
158                }
159
160                $sql = "UPDATE {$table} SET {$set} WHERE `".implode('` = `', $whereKeys)."`";
161
162                if(!$this->query($sql, $fields)->error()){
163                    return true;
164                }
165            }
166        }
167
168        0 references | 0 overrides
169        public function delete($table, $where){
170            if(count($where)){
171                $sql = "DELETE FROM {$table} WHERE `".implode('` = `', $where)."`";
172
173                if(!$this->query($sql)->error()){
174                    return true;
175                }
176            }
177        }
178
179        0 references | 0 overrides
180        public function truncate($table){
181            $sql = "TRUNCATE TABLE {$table}";
182
183            if(!$this->query($sql)->error()){
184                return true;
185            }
186        }
187
188        0 references | 0 overrides
189        public function drop($table){
190            $sql = "DROP TABLE {$table}";
191
192            if(!$this->query($sql)->error()){
193                return true;
194            }
195        }
196
197        0 references | 0 overrides
198        public function truncateTable($table){
199            $sql = "TRUNCATE TABLE {$table}";
200
201            if(!$this->query($sql)->error()){
202                return true;
203            }
204        }
205
206        0 references | 0 overrides
207        public function dropTable($table){
208            $sql = "DROP TABLE {$table}";
209
210            if(!$this->query($sql)->error()){
211                return true;
212            }
213        }
214
215        0 references | 0 overrides
216        public function dropDatabase($database){
217            $sql = "DROP DATABASE {$database}";
218
219            if(!$this->query($sql)->error()){
220                return true;
221            }
222        }
223
224        0 references | 0 overrides
225        public function createTable($table, $fields, $key = null, $type = null, $collation = null, $comment = null){
226            $sql = "CREATE TABLE {$table} (";
227
228            foreach($fields as $field){
229                $sql .= "{$field} ";
230                if($type){
231                    $sql .= "{$type} ";
232                }
233                if($collation){
234                    $sql .= "{$collation} ";
235                }
236                if($comment){
237                    $sql .= "{$comment} ";
238                }
239            }
240
241            if($key){
242                $sql .= "PRIMARY KEY(`{$key}`)";
243            }
244
245            $sql .= ")";
246
247            if(!$this->query($sql)->error()){
248                return true;
249            }
250        }
251
252        0 references | 0 overrides
253        public function createDatabase($database){
254            $sql = "CREATE DATABASE {$database}";
255
256            if(!$this->query($sql)->error()){
257                return true;
258            }
259        }
260
261        0 references | 0 overrides
262        public function dropDatabase($database){
263            $sql = "DROP DATABASE {$database}";
264
265            if(!$this->query($sql)->error()){
266                return true;
267            }
268        }
269
270        0 references | 0 overrides
271        public function createTableIfNotExists($table, $fields, $key = null, $type = null, $collation = null, $comment = null){
272            $sql = "CREATE TABLE IF NOT EXISTS {$table} (";
273
274            foreach($fields as $field){
275                $sql .= "{$field} ";
276                if($type){
277                    $sql .= "{$type} ";
278                }
279                if($collation){
280                    $sql .= "{$collation} ";
281                }
282                if($comment){
283                    $sql .= "{$comment} ";
284                }
285            }
286
287            if($key){
288                $sql .= "PRIMARY KEY(`{$key}`)";
289            }
290
291            $sql .= ")";
292
293            if(!$this->query($sql)->error()){
294                return true;
295            }
296        }
297
298        0 references | 0 overrides
299        public function dropTableIfExists($table){
300            $sql = "DROP TABLE IF EXISTS {$table}";
301
302            if(!$this->query($sql)->error()){
303                return true;
304            }
305        }
306
307        0 references | 0 overrides
308        public function dropDatabaseIfExists($database){
309            $sql = "DROP DATABASE IF EXISTS {$database}";
310
311            if(!$this->query($sql)->error()){
312                return true;
313            }
314        }
315
316        0 references | 0 overrides
317        public function truncateTableIfExists($table){
318            $sql = "TRUNCATE TABLE IF EXISTS {$table}";
319
320            if(!$this->query($sql)->error()){
321                return true;
322            }
323        }
324
325        0 references | 0 overrides
326        public function dropTable($table, $key = null, $type = null, $collation = null, $comment = null){
327            $sql = "DROP TABLE {$table} ";
328
329            if($key){
330                $sql .= "PRIMARY KEY(`{$key}`)";
331            }
332
333            if($type){
334                $sql .= "{$type} ";
335            }
336            if($collation){
337                $sql .= "{$collation} ";
338            }
339            if($comment){
340                $sql .= "{$comment} ";
341            }
342
343            if(!$this->query($sql)->error()){
344                return true;
345            }
346        }
347
348        0 references | 0 overrides
349        public function dropDatabase($database, $key = null, $type = null, $collation = null, $comment = null){
350            $sql = "DROP DATABASE {$database} ";
351
352            if($key){
353                $sql .= "PRIMARY KEY(`{$key}`)";
354            }
355
356            if($type){
357                $sql .= "{$type} ";
358            }
359            if($collation){
360                $sql .= "{$collation} ";
361            }
362            if($comment){
363                $sql .= "{$comment} ";
364            }
365
366            if(!$this->query($sql)->error()){
367                return true;
368            }
369        }
370
371        0 references | 0 overrides
372        public function dropTableOrDatabase($tableOrDatabase, $key = null, $type = null, $collation = null, $comment = null){
373            $sql = "DROP {$tableOrDatabase} ";
374
375            if($key){
376                $sql .= "PRIMARY KEY(`{$key}`)";
377            }
378
379            if($type){
380                $sql .= "{$type} ";
381            }
382            if($collation){
383                $sql .= "{$collation} ";
384            }
385            if($comment){
386                $sql .= "{$comment} ";
387            }
388
389            if(!$this->query($sql)->error()){
390                return true;
391            }
392        }
393
394        0 references | 0 overrides
395        public function dropTableOrDatabaseIfExists($tableOrDatabase, $key = null, $type = null, $collation = null, $comment = null){
396            $sql = "DROP {$tableOrDatabase} IF EXISTS ";
397
398            if($key){
399                $sql .= "PRIMARY KEY(`{$key}`)";
400            }
401
402            if($type){
403                $sql .= "{$type} ";
404            }
405            if($collation){
406                $sql .= "{$collation} ";
407            }
408            if($comment){
409                $sql .= "{$comment} ";
410            }
411
412            if(!$this->query($sql)->error()){
413                return true;
414            }
415        }
416
417        0 references | 0 overrides
418        public function dropTableOrDatabaseOrTable($tableOrDatabaseOrTable, $key = null, $type = null, $collation = null, $comment = null){
419            $sql = "DROP {$tableOrDatabaseOrTable} ";
420
421            if($key){
422                $sql .= "PRIMARY KEY(`{$key}`)";
423            }
424
425            if($type){
426                $sql .= "{$type} ";
427            }
428            if($collation){
429                $sql .= "{$collation} ";
430            }
431            if($comment){
432                $sql .= "{$comment} ";
433            }
434
435            if(!$this->query($sql)->error()){
436                return true;
437            }
438        }
439
440        0 references | 0 overrides
441        public function dropTableOrDatabaseOrTableIfExists($tableOrDatabaseOrTable, $key = null, $type = null, $collation = null, $comment = null){
442            $sql = "DROP {$tableOrDatabaseOrTable} IF EXISTS ";
443
444            if($key){
445                $sql .= "PRIMARY KEY(`{$key}`)";
446            }
447
448            if($type){
449                $sql .= "{$type} ";
450            }
451            if($collation){
452                $sql .= "{$collation} ";
453            }
454            if($comment){
455                $sql .= "{$comment} ";
456            }
457
458            if(!$this->query($sql)->error()){
459                return true;
460            }
461        }
462
463        0 references | 0 overrides
464        public function dropTableOrDatabaseOrTableOrTable($tableOrDatabaseOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
465            $sql = "DROP {$tableOrDatabaseOrTableOrTable} ";
466
467            if($key){
468                $sql .= "PRIMARY KEY(`{$key}`)";
469            }
470
471            if($type){
472                $sql .= "{$type} ";
473            }
474            if($collation){
475                $sql .= "{$collation} ";
476            }
477            if($comment){
478                $sql .= "{$comment} ";
479            }
480
481            if(!$this->query($sql)->error()){
482                return true;
483            }
484        }
485
486        0 references | 0 overrides
487        public function dropTableOrDatabaseOrTableOrTableIfExists($tableOrDatabaseOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
488            $sql = "DROP {$tableOrDatabaseOrTableOrTable} IF EXISTS ";
489
490            if($key){
491                $sql .= "PRIMARY KEY(`{$key}`)";
492            }
493
494            if($type){
495                $sql .= "{$type} ";
496            }
497            if($collation){
498                $sql .= "{$collation} ";
499            }
500            if($comment){
501                $sql .= "{$comment} ";
502            }
503
504            if(!$this->query($sql)->error()){
505                return true;
506            }
507        }
508
509        0 references | 0 overrides
510        public function dropTableOrDatabaseOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
511            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTable} ";
512
513            if($key){
514                $sql .= "PRIMARY KEY(`{$key}`)";
515            }
516
517            if($type){
518                $sql .= "{$type} ";
519            }
520            if($collation){
521                $sql .= "{$collation} ";
522            }
523            if($comment){
524                $sql .= "{$comment} ";
525            }
526
527            if(!$this->query($sql)->error()){
528                return true;
529            }
530        }
531
532        0 references | 0 overrides
533        public function dropTableOrDatabaseOrTableOrTableOrTableIfExists($tableOrDatabaseOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
534            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTable} IF EXISTS ";
535
536            if($key){
537                $sql .= "PRIMARY KEY(`{$key}`)";
538            }
539
540            if($type){
541                $sql .= "{$type} ";
542            }
543            if($collation){
544                $sql .= "{$collation} ";
545            }
546            if($comment){
547                $sql .= "{$comment} ";
548            }
549
550            if(!$this->query($sql)->error()){
551                return true;
552            }
553        }
554
555        0 references | 0 overrides
556        public function dropTableOrDatabaseOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
557            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTable} ";
558
559            if($key){
560                $sql .= "PRIMARY KEY(`{$key}`)";
561            }
562
563            if($type){
564                $sql .= "{$type} ";
565            }
566            if($collation){
567                $sql .= "{$collation} ";
568            }
569            if($comment){
570                $sql .= "{$comment} ";
571            }
572
573            if(!$this->query($sql)->error()){
574                return true;
575            }
576        }
577
578        0 references | 0 overrides
579        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableIfExists($tableOrDatabaseOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
580            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTable} IF EXISTS ";
581
582            if($key){
583                $sql .= "PRIMARY KEY(`{$key}`)";
584            }
585
586            if($type){
587                $sql .= "{$type} ";
588            }
589            if($collation){
590                $sql .= "{$collation} ";
591            }
592            if($comment){
593                $sql .= "{$comment} ";
594            }
595
596            if(!$this->query($sql)->error()){
597                return true;
598            }
599        }
600
601        0 references | 0 overrides
602        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
603            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTable} ";
604
605            if($key){
606                $sql .= "PRIMARY KEY(`{$key}`)";
607            }
608
609            if($type){
610                $sql .= "{$type} ";
611            }
612            if($collation){
613                $sql .= "{$collation} ";
614            }
615            if($comment){
616                $sql .= "{$comment} ";
617            }
618
619            if(!$this->query($sql)->error()){
620                return true;
621            }
622        }
623
624        0 references | 0 overrides
625        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableIfExists($tableOrDatabaseOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
626            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTable} IF EXISTS ";
627
628            if($key){
629                $sql .= "PRIMARY KEY(`{$key}`)";
630            }
631
632            if($type){
633                $sql .= "{$type} ";
634            }
635            if($collation){
636                $sql .= "{$collation} ";
637            }
638            if($comment){
639                $sql .= "{$comment} ";
640            }
641
642            if(!$this->query($sql)->error()){
643                return true;
644            }
645        }
646
647        0 references | 0 overrides
648        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
649            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTable} ";
650
651            if($key){
652                $sql .= "PRIMARY KEY(`{$key}`)";
653            }
654
655            if($type){
656                $sql .= "{$type} ";
657            }
658            if($collation){
659                $sql .= "{$collation} ";
660            }
661            if($comment){
662                $sql .= "{$comment} ";
663            }
664
665            if(!$this->query($sql)->error()){
666                return true;
667            }
668        }
669
670        0 references | 0 overrides
671        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableIfExists($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
672            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTable} IF EXISTS ";
673
674            if($key){
675                $sql .= "PRIMARY KEY(`{$key}`)";
676            }
677
678            if($type){
679                $sql .= "{$type} ";
680            }
681            if($collation){
682                $sql .= "{$collation} ";
683            }
684            if($comment){
685                $sql .= "{$comment} ";
686            }
687
688            if(!$this->query($sql)->error()){
689                return true;
690            }
691        }
692
693        0 references | 0 overrides
694        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
695            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
696
697            if($key){
698                $sql .= "PRIMARY KEY(`{$key}`)";
699            }
700
701            if($type){
702                $sql .= "{$type} ";
703            }
704            if($collation){
705                $sql .= "{$collation} ";
706            }
707            if($comment){
708                $sql .= "{$comment} ";
709            }
710
711            if(!$this->query($sql)->error()){
712                return true;
713            }
714        }
715
716        0 references | 0 overrides
717        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
718            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
719
720            if($key){
721                $sql .= "PRIMARY KEY(`{$key}`)";
722            }
723
724            if($type){
725                $sql .= "{$type} ";
726            }
727            if($collation){
728                $sql .= "{$collation} ";
729            }
730            if($comment){
731                $sql .= "{$comment} ";
732            }
733
734            if(!$this->query($sql)->error()){
735                return true;
736            }
737        }
738
739        0 references | 0 overrides
740        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
741            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
742
743            if($key){
744                $sql .= "PRIMARY KEY(`{$key}`)";
745            }
746
747            if($type){
748                $sql .= "{$type} ";
749            }
750            if($collation){
751                $sql .= "{$collation} ";
752            }
753            if($comment){
754                $sql .= "{$comment} ";
755            }
756
757            if(!$this->query($sql)->error()){
758                return true;
759            }
760        }
761
762        0 references | 0 overrides
763        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
764            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
765
766            if($key){
767                $sql .= "PRIMARY KEY(`{$key}`)";
768            }
769
770            if($type){
771                $sql .= "{$type} ";
772            }
773            if($collation){
774                $sql .= "{$collation} ";
775            }
776            if($comment){
777                $sql .= "{$comment} ";
778            }
779
780            if(!$this->query($sql)->error()){
781                return true;
782            }
783        }
784
785        0 references | 0 overrides
786        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
787            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
788
789            if($key){
790                $sql .= "PRIMARY KEY(`{$key}`)";
791            }
792
793            if($type){
794                $sql .= "{$type} ";
795            }
796            if($collation){
797                $sql .= "{$collation} ";
798            }
799            if($comment){
800                $sql .= "{$comment} ";
801            }
802
803            if(!$this->query($sql)->error()){
804                return true;
805            }
806        }
807
808        0 references | 0 overrides
809        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
810            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
811
812            if($key){
813                $sql .= "PRIMARY KEY(`{$key}`)";
814            }
815
816            if($type){
817                $sql .= "{$type} ";
818            }
819            if($collation){
820                $sql .= "{$collation} ";
821            }
822            if($comment){
823                $sql .= "{$comment} ";
824            }
825
826            if(!$this->query($sql)->error()){
827                return true;
828            }
829        }
830
831        0 references | 0 overrides
832        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
833            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
834
835            if($key){
836                $sql .= "PRIMARY KEY(`{$key}`)";
837            }
838
839            if($type){
840                $sql .= "{$type} ";
841            }
842            if($collation){
843                $sql .= "{$collation} ";
844            }
845            if($comment){
846                $sql .= "{$comment} ";
847            }
848
849            if(!$this->query($sql)->error()){
850                return true;
851            }
852        }
853
854        0 references | 0 overrides
855        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
856            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
857
858            if($key){
859                $sql .= "PRIMARY KEY(`{$key}`)";
860            }
861
862            if($type){
863                $sql .= "{$type} ";
864            }
865            if($collation){
866                $sql .= "{$collation} ";
867            }
868            if($comment){
869                $sql .= "{$comment} ";
870            }
871
872            if(!$this->query($sql)->error()){
873                return true;
874            }
875        }
876
877        0 references | 0 overrides
878        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
879            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
880
881            if($key){
882                $sql .= "PRIMARY KEY(`{$key}`)";
883            }
884
885            if($type){
886                $sql .= "{$type} ";
887            }
888            if($collation){
889                $sql .= "{$collation} ";
890            }
891            if($comment){
892                $sql .= "{$comment} ";
893            }
894
895            if(!$this->query($sql)->error()){
896                return true;
897            }
898        }
899
900        0 references | 0 overrides
901        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
902            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
903
904            if($key){
905                $sql .= "PRIMARY KEY(`{$key}`)";
906            }
907
908            if($type){
909                $sql .= "{$type} ";
910            }
911            if($collation){
912                $sql .= "{$collation} ";
913            }
914            if($comment){
915                $sql .= "{$comment} ";
916            }
917
918            if(!$this->query($sql)->error()){
919                return true;
920            }
921        }
922
923        0 references | 0 overrides
924        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
925            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
926
927            if($key){
928                $sql .= "PRIMARY KEY(`{$key}`)";
929            }
930
931            if($type){
932                $sql .= "{$type} ";
933            }
934            if($collation){
935                $sql .= "{$collation} ";
936            }
937            if($comment){
938                $sql .= "{$comment} ";
939            }
940
941            if(!$this->query($sql)->error()){
942                return true;
943            }
944        }
945
946        0 references | 0 overrides
947        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
948            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
949
950            if($key){
951                $sql .= "PRIMARY KEY(`{$key}`)";
952            }
953
954            if($type){
955                $sql .= "{$type} ";
956            }
957            if($collation){
958                $sql .= "{$collation} ";
959            }
960            if($comment){
961                $sql .= "{$comment} ";
962            }
963
964            if(!$this->query($sql)->error()){
965                return true;
966            }
967        }
968
969        0 references | 0 overrides
970        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
971            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
972
973            if($key){
974                $sql .= "PRIMARY KEY(`{$key}`)";
975            }
976
977            if($type){
978                $sql .= "{$type} ";
979            }
980            if($collation){
981                $sql .= "{$collation} ";
982            }
983            if($comment){
984                $sql .= "{$comment} ";
985            }
986
987            if(!$this->query($sql)->error()){
988                return true;
989            }
990        }
991
992        0 references | 0 overrides
993        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
994            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
995
996            if($key){
997                $sql .= "PRIMARY KEY(`{$key}`)";
998            }
999
1000            if($type){
1001                $sql .= "{$type} ";
1002            }
1003            if($collation){
1004                $sql .= "{$collation} ";
1005            }
1006            if($comment){
1007                $sql .= "{$comment} ";
1008            }
1009
1010            if(!$this->query($sql)->error()){
1011                return true;
1012            }
1013        }
1014
1015        0 references | 0 overrides
1016        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
1017            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
1018
1019            if($key){
1020                $sql .= "PRIMARY KEY(`{$key}`)";
1021            }
1022
1023            if($type){
1024                $sql .= "{$type} ";
1025            }
1026            if($collation){
1027                $sql .= "{$collation} ";
1028            }
1029            if($comment){
1030                $sql .= "{$comment} ";
1031            }
1032
1033            if(!$this->query($sql)->error()){
1034                return true;
1035            }
1036        }
1037
1038        0 references | 0 overrides
1039        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
1040            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
1041
1042            if($key){
1043                $sql .= "PRIMARY KEY(`{$key}`)";
1044            }
1045
1046            if($type){
1047                $sql .= "{$type} ";
1048            }
1049            if($collation){
1050                $sql .= "{$collation} ";
1051            }
1052            if($comment){
1053                $sql .= "{$comment} ";
1054            }
1055
1056            if(!$this->query($sql)->error()){
1057                return true;
1058            }
1059        }
1060
1061        0 references | 0 overrides
1062        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
1063            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
1064
1065            if($key){
1066                $sql .= "PRIMARY KEY(`{$key}`)";
1067            }
1068
1069            if($type){
1070                $sql .= "{$type} ";
1071            }
1072            if($collation){
1073                $sql .= "{$collation} ";
1074            }
1075            if($comment){
1076                $sql .= "{$comment} ";
1077            }
1078
1079            if(!$this->query($sql)->error()){
1080                return true;
1081            }
1082        }
1083
1084        0 references | 0 overrides
1085        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
1086            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
1087
1088            if($key){
1089                $sql .= "PRIMARY KEY(`{$key}`)";
1090            }
1091
1092            if($type){
1093                $sql .= "{$type} ";
1094            }
1095            if($collation){
1096                $sql .= "{$collation} ";
1097            }
1098            if($comment){
1099                $sql .= "{$comment} ";
1100            }
1101
1102            if(!$this->query($sql)->error()){
1103                return true;
1104            }
1105        }
1106
1107        0 references | 0 overrides
1108        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
1109            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
1110
1111            if($key){
1112                $sql .= "PRIMARY KEY(`{$key}`)";
1113            }
1114
1115            if($type){
1116                $sql .= "{$type} ";
1117            }
1118            if($collation){
1119                $sql .= "{$collation} ";
1120            }
1121            if($comment){
1122                $sql .= "{$comment} ";
1123            }
1124
1125            if(!$this->query($sql)->error()){
1126                return true;
1127            }
1128        }
1129
1130        0 references | 0 overrides
1131        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
1132            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
1133
1134            if($key){
1135                $sql .= "PRIMARY KEY(`{$key}`)";
1136            }
1137
1138            if($type){
1139                $sql .= "{$type} ";
1140            }
1141            if($collation){
1142                $sql .= "{$collation} ";
1143            }
1144            if($comment){
1145                $sql .= "{$comment} ";
1146            }
1147
1148            if(!$this->query($sql)->error()){
1149                return true;
1150            }
1151        }
1152
1153        0 references | 0 overrides
1154        public function dropTableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable($tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable, $key = null, $type = null, $collation = null, $comment = null){
1155            $sql = "DROP {$tableOrDatabaseOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTableOrTable} ";
1156
1157            if($key){
1158                $sql .= "PRIMARY KEY(`{$key}`)";
1159            }
1160
1161            if($type){
1162                $sql .= "{$type} ";
1163            }
1164            if($collation){
1165                $sql .= "{$collation} ";
1166            }
1167            if($comment){
1168                $sql .= "{$comment} ";

```

```

database.php ×

C: > xampp > htdocs > ezgo > database.php > PHP > DB > update()

  2   class DB{
143     public function update($table, $fields, $where){
157
158       $whereClause = null;
159       foreach ($whereKeys as $key) {
160         $whereClause .= "$key = ? AND ";
161       }
162       $whereClause = rtrim($whereClause, ' AND ');
163
164
165       $sql = "UPDATE {$table} SET {$set} WHERE {$whereClause}";
166
167       $fields = array_merge(array_values($fields), array_values($where));
168
169       if (!$this->query($sql, $fields)->error()) {
170         return true;
171       }
172     }
173     return false;
174   }
175
176   1 reference | 0 overrides
177   public function delete($table, $where){
178     $sql = "DELETE FROM {$table} WHERE ";
179     $value = array();
180
181     if($this->isRectangular($where)){
182       $x=1;
183       for($i = 0; $i < count($where); $i++){
184         $sql .= $where[$i][0]." ".$where[$i][1]." ?";
185         if($x < count($where)){
186           $sql.=" AND ";
187         }
188         $value[] = $where[$i][2];
189         $x++;
190       }
191     }else{
192       $sql .= $where[0]." ".$where[1]." ?";
193       $value[] = $where[2];
194     }
195
196     if(!$this->query($sql, $value)->error()){
197       return true;
198     }
199     return false;
200   }

```

Kode PHP database di atas merupakan implementasi dari kelas *DB* yang menyediakan metode untuk berinteraksi dengan database menggunakan PDO (PHP Data Objects). Kelas ini mengimplementasikan pola singleton untuk memastikan hanya ada satu instance yang dibuat. Fungsi utama yang disediakan termasuk *query* untuk menjalankan query SQL, *select* untuk mengambil data dari tabel, *insert* untuk menambah data ke tabel, *update* untuk memperbarui data

dalam tabel, dan *delete* untuk menghapus data dari tabel. Kelas ini juga memiliki metode untuk memeriksa kesalahan dan mendapatkan hasil query. Fitur-fitur ini penting untuk memudahkan operasi CRUD (Create, Read, Update, Delete) di dalam aplikasi, dengan mengenkapsulasi logika database sehingga lebih mudah dikelola dan digunakan dalam berbagai bagian aplikasi.

### 3.3.4. exploreController.php



```
C: > xampp > htdocs > ezgo > exploreController.php > ...
1  <?php
2
3  namespace App\Controller;
4
5  use App\Tools\DB;
6
7  class explore{
8      0 references | 0 overrides
9      public function getAll(){
10         $db = DB::getInstance();
11
12         $locs = $db->select('*', 'locations')->getResult();
13
14         if(count($locs) > 0){
15             foreach($locs as $loc){
16                 $imgPath = $loc->lImage;
17                 $fullPath = "images/".$imgPath;
18                 $loc->lImage = $fullPath;
19             }
20
21             $cities = $db->select('*', 'city')->getResult();
22             if(count($cities) > 0){
23                 $response = ['Success'=> True, 'locations' => $locs, 'cities' => $cities];
24                 header('Content-Type: application/json');
25                 echo json_encode($response);
26             }else{
27                 $response = ['Success'=> False];
28                 header('Content-Type: application/json');
29                 echo json_encode($response);
30             }
31         }else{
32             $response = ['Success'=> False];
33             header('Content-Type: application/json');
34             echo json_encode($response);
35         }
36     }
37 ?>
```

Kode PHP exploreController di atas merupakan bagian dari controller yang mengelola eksplorasi lokasi dan kota dalam suatu aplikasi. Controller ini memiliki satu fungsi utama yaitu `getAll()`, yang digunakan untuk mengambil semua data lokasi dari tabel *locations* dan semua data kota dari tabel *city* dalam database. Setelah mengambil data lokasi, kode ini memperbarui jalur gambar lokasi dengan menambahkan prefix "images/" ke nama file gambar. Jika data lokasi dan data kota berhasil diambil dari database, controller ini akan mengembalikan *respons JSON* yang berisi data lokasi dan data kota. Jika salah satu atau kedua data tersebut tidak ditemukan, *respons JSON* yang dikembalikan akan menunjukkan kegagalan. Kode ini mengimplementasikan fitur eksplorasi lokasi dan kota, yang penting untuk aplikasi yang menyediakan informasi tempat-tempat menarik atau tujuan wisata.

### ***3.3.5. locationController.php***

```
locationController.php ×
C: > xampp > htdocs > ezgo > locationController.php > PHP > location > like()
1  <?php
2  include_once 'database.php';
3
4  class location{
5      0 references | 0 implementations
6      0 references | 0 overrides
7      public function homePopular(){
8          $db = DB::getInstance();
9
10         $locs = $db->select('*', 'locations', null, 6, '1Likes', 'DESC')->getResult();
11
12         if(count($locs) > 0){
13             $response = ['Success'=> True, 'Data' => $locs];
14             header('Content-Type: application/json');
15             echo json_encode($response);
16         }else{
17             $response = ['Success'=> False];
18             header('Content-Type: application/json');
19             echo json_encode($response);
20         }
21
22         0 references | 0 overrides
23         public function explore(){
24             $db = DB::getInstance();
25
26             $locs = $db->select('*', 'locations', null, null, '1Likes', 'DESC')->getResult();
27
28             if(count($locs) > 0){
29                 $response = ['Success'=> True, 'Data' => $locs];
30                 header('Content-Type: application/json');
31                 echo json_encode($response);
32             }else{
33                 $response = ['Success'=> False];
34                 header('Content-Type: application/json');
35                 echo json_encode($response);
36             }
37
38             0 references | 0 overrides
39             public function like(){}
```

```
locationController.php X
C: > xampp > htdocs > ezgo > locationController.php > ...
4   class location{
38     public function like(){
39       $db = DB::getInstance();
40       $json = file_get_contents('php://input');
41
42       $request = json_decode($json, true);
43       $lid = $request['locID'];
44       $like = $request['lLikes'];
45
46       $fields = array(
47         'lLikes' => $like
48       );
49       $where = array(
50         'locID' => $lid
51       );
52       $update = $db->update('locations', $fields, $where);
53       if($update){
54         $response = ['Success'=> True];
55         header('Content-Type: application/json');
56         echo json_encode($response);
57       }else{
58         $response = ['Success'=> False];
59         header('Content-Type: application/json');
60         echo json_encode($response);
61       }
62     }
63   }
64 ?>
```

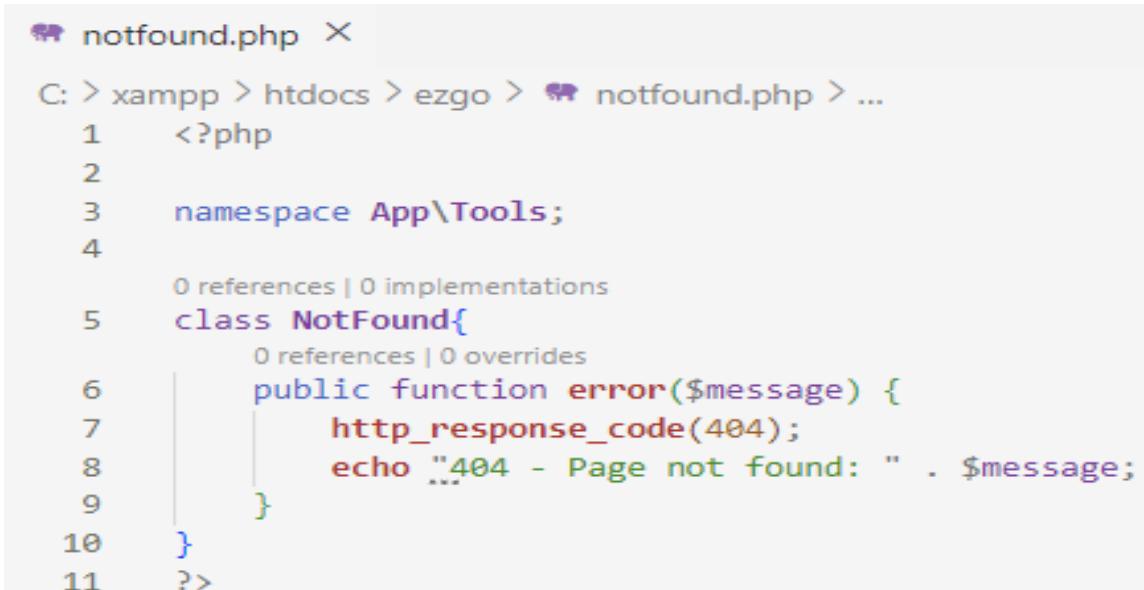
Kode PHP locationController di atas merupakan bagian dari controller yang mengelola data lokasi dalam suatu aplikasi. Controller ini menyediakan tiga fungsi utama: *homePopular()*, *explore()*, dan *like()*. Fungsi *homePopular()* digunakan untuk mengambil enam lokasi populer berdasarkan jumlah likes secara menurun (descending) dari tabel *locations*. Fungsi *explore()* mengambil semua lokasi dari tabel *locations* yang diurutkan berdasarkan jumlah likes secara menurun. Fungsi *like()* digunakan untuk memperbarui jumlah likes dari suatu lokasi berdasarkan *locID* yang diterima dari permintaan HTTP. Semua fungsi ini berinteraksi dengan database menggunakan kelas *DB* untuk melakukan operasi pengambilan dan pembaruan data. Kode ini mengimplementasikan fitur eksplorasi lokasi populer dan pembaruan likes, yang penting untuk aplikasi yang menawarkan informasi mengenai tempat-tempat menarik atau tujuan wisata yang banyak diminati pengguna.

### 3.3.6. loginController.php

```
loginController.php X
C: > xampp > htdocs > ezgo > loginController.php > ...
4     class login{
43         public function login(){
53             if(count($user) > 0){
54                 if(strcasecmp($pw, $user[0]->uPassword) == 0){
55                     $response = ['Success' => True, 'Data' => $user[0]];
56                     header('Content-Type: application/json');
57                     echo json_encode($response);
58                 }else{
59                     $response = ['Success' => False];
60                     header('Content-Type: application/json');
61                     echo json_encode($response);
62                 }
63             }else{
64                 $response = ['Success' => False];
65                 header('Content-Type: application/json');
66                 echo json_encode($response);
67             }
68         }
69
70         0 references | 0 overrides
70         public function createAccount(){
71             $db = DB::getInstance();
72             $json = file_get_contents('php://input');
73
74             $request = json_decode($json, true);
75             $uid = $request['userID'];
76             $pw = $request['uPassword'];
77             $name = $request['uName'];
78             $mail = $request['uEmail'];
79             $telp = $request['uPhone'];
80
81             $fields = array('userID', 'uPassword', 'uName', 'uEmail', 'uPhone');
82             $values = array($uid, $pw, $name, $mail, $telp);
83             $assoc_fields = array_combine($fields, $values);
84
85             try{
86                 if($db->insert('users', $assoc_fields)){
87                     $response = ['Success'=> True, 'Message' => 1];
88                     header('Content-Type: application/json');
89                     echo json_encode($response);
90                 }else{
91                     $response = ['Success'=> False, 'Message' => 2];
92                     header('Content-Type: application/json');
93                     echo json_encode($response);
94                 }
95             }catch(Exception $e){
96                 $response = ['Success'=> False, 'Message' => 3];
97                 header('Content-Type: application/json');
98                 echo json_encode($response);
99             }
100         }
101     }
102 ?>
```

Kode PHP loginController di atas merupakan bagian dari controller yang mengelola proses login dan pembuatan akun dalam suatu aplikasi. Controller ini menyediakan empat fungsi utama: *checkID()*, *checkExist()*, *login()*, dan *createAccount()*. Fungsi *checkID()* dan *checkExist()* digunakan untuk memeriksa keberadaan *userID* dalam database, dimana *checkID()* mengembalikan true jika *userID* ditemukan dan *checkExist()* mengembalikan true jika *userID* tidak ditemukan. Fungsi *login()* memverifikasi kredensial pengguna dengan mencocokkan *userID* dan *uPassword* yang diterima dari permintaan HTTP dengan data dalam database. Fungsi *createAccount()* digunakan untuk membuat akun baru dengan menyimpan informasi pengguna seperti *userID*, *uPassword*, *uName*, *uEmail*, dan *uPhone* ke dalam database. Semua fungsi ini berinteraksi dengan database menggunakan kelas *DB* untuk melakukan operasi validasi dan manajemen data pengguna. Kode ini mengimplementasikan fitur-fitur penting terkait autentikasi dan pembuatan akun untuk mengamankan akses dan mengelola pengguna dalam aplikasi.

### 3.3.7. notfound.php



```
notfound.php ×
C: > xampp > htdocs > ezgo > notfound.php > ...
1  <?php
2
3  namespace App\Tools;
4
5  class NotFound{
6      0 references | 0 implementations
7      0 references | 0 overrides
8      public function error($message) {
9          http_response_code(404);
10         echo "404 - Page not found: " . $message;
11     }
12 ?>
```

Kode PHP notfound di atas merupakan bagian dari kelas *NotFound* yang berada di dalam namespace *App\Tools*. Kelas ini memiliki satu fungsi utama, yaitu *error(\$message)*, yang digunakan untuk mengirimkan respons HTTP dengan kode status 404 (Not Found) beserta pesan yang diberikan sebagai parameter. Fungsi ini digunakan untuk menangani dan memberikan respons ketika halaman atau sumber daya yang diminta oleh pengguna tidak ditemukan dalam aplikasi. Dengan mengimplementasikan fitur ini, aplikasi dapat memberikan pesan kesalahan

yang informatif kepada pengguna ketika mereka mencoba mengakses URL atau halaman yang tidak ada, sehingga meningkatkan pengalaman pengguna dan membantu dalam debugging.

### ***3.3.8. orderController.php***

```
orderController.php 5 ×
C: > xampp > htdocs > ezgo > orderController.php > PHP > order > getMixedProd()
1  <?php
2  include_once 'database.php';
3
4  class order{
5      0 references | 0 implementations
6      0 references | 0 overrides
7      public function orderTicket(){
8          $db = DB::getInstance();
9          $json = file_get_contents('php://input');
10
11         $request = json_decode($json, true);
12         $tid = $request['productID'];
13         $uid = $request['userID'];
14         $pmt = $request['payMethod'];
15         $amt = $request['tdAmount'];
16         $ttp = $request['tdTotalPrice'];
17         $id = $request['id'];
18         $uamt = $request['upAmount'];
19
20         $fields = array('userID', 'paymentMethod', 'productID', 'tdAmount', 'tdTotalPrice');
21         $values = array($uid, $pmt, $tid, $amt, $ttp);
22         $assoc_fields = array_combine($fields, $values);
23
24         if($db->insert('transaction_details', $assoc_fields)){
25             $fields = array('tseat');
26             $values = array($uamt);
27             $assoc_array = array_combine($fields, $values);
28
29             $where = array('ticketID', $id);
30
31             if($db->update('tickets', $assoc_array, $where)){
32                 $response = ['Success'=> True];
33                 header('Content-Type: application/json');
34                 echo json_encode($response);
35             }else{
36                 $response = ['Success'=> False];
37                 header('Content-Type: application/json');
38                 echo json_encode($response);
39             }
40         }else{
41             $response = ['Success'=> False];
42             header('Content-Type: application/json');
43             echo json_encode($response);
44         }
45
46         0 references | 0 overrides
47         public function orderHotel(){
48             $db = DB::getInstance();
49             $json = file_get_contents('php://input');
50
51             $request = json_decode($json, true);
52             $tid = $request['productID'];
53             $uid = $request['userID'];
54             $pmt = $request['payMethod'];
55             $amt = $request['tdAmount'];
56             $ttp = $request['tdTotalPrice'];
```

```
orderController.php 5 X
C: > xampp > htdocs > ezgo > orderController.php > PHP > order > orderTicket()
  4  class order{
  5      public function orderHotel(){
  6
  7          $id = $request['id'];
  8          $uamt = $request['upAmount'];
  9
 10
 11          $fields = array('userID', 'paymentMethod', 'productID', 'tdAmount', 'tdTotalPrice');
 12          $values = array($uid, $pmt, $tid, $amt, $ttp);
 13          $assoc_fields = array_combine($fields, $values);
 14
 15
 16          if($db->insert('transaction_details', $assoc_fields)){
 17              $fields = array('hNights');
 18              $values = array($uamt);
 19              $assoc_array = array_combine($fields, $values);
 20
 21
 22              $where = array('hotelID', $id);
 23
 24
 25              if($db->update('hotel', $assoc_array, $where)){
 26                  $response = ['Success'=> True];
 27                  header('Content-Type: application/json');
 28                  echo json_encode($response);
 29              }else{
 30                  $response = ['Success'=> False];
 31                  header('Content-Type: application/json');
 32                  echo json_encode($response);
 33              }
 34          }else{
 35              $response = ['Success'=> False];
 36              header('Content-Type: application/json');
 37              echo json_encode($response);
 38          }
 39      }
 40
 41
 42      0 references | 0 overrides
 43      public function orderTour(){
 44          $db = DB::getInstance();
 45          $json = file_get_contents('php://input');
 46
 47
 48          $request = json_decode($json, true);
 49          $tid = $request['productID'];
 50          $uid = $request['userID'];
 51          $pmt = $request['payMethod'];
 52          $amt = $request['tdAmount'];
 53          $ttp = $request['tdTotalPrice'];
 54          $id = $request['id'];
 55          $uamt = $request['upAmount'];
 56
 57
 58          $fields = array('userID', 'paymentMethod', 'productID', 'tdAmount', 'tdTotalPrice');
 59          $values = array($uid, $pmt, $tid, $amt, $ttp);
 60          $assoc_fields = array_combine($fields, $values);
 61
 62
 63          if($db->insert('transaction_details', $assoc_fields)){
 64              $fields = array('tPSlot');
 65              $values = array($uamt);
 66              $assoc_array = array_combine($fields, $values);
 67
 68              $where = array('tourID', $id);
 69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
100
101
102
103
104
105
106
107
```

```
orderController.php 5 ×

C: > xampp > htdocs > ezgo > orderController.php > PHP > order > orderTour()
  4     class order{
  85         public function orderTour(){
108             if($db->update('tour_package', $assoc_array, $where)){
110                 $response = ['Success'=> True];
111                 header('Content-Type: application/json');
112                 echo json_encode($response);
113             }else{
114                 $response = ['Success'=> False];
115                 header('Content-Type: application/json');
116                 echo json_encode($response);
117             }
118         }else{
119             $response = ['Success'=> False];
120             header('Content-Type: application/json');
121             echo json_encode($response);
122         }
123     }
124
125     0 references | 0 overrides
126     public function changeTicket(){
127         $db = DB::getInstance();
128         $json = file_get_contents('php://input');
129
130         $request = json_decode($json, true);
131         $tix = request['ticket'];
132
133         $fields = array('tcSeat');
134         $values = array($tix->tcSeat);
135         $assoc_array = array_combine($fields, $values);
136
137         $where = array('ticketID', $tix->ticketID);
138
139         if($db->update('tickets', $assoc_array, $where)){
140             $response = ['Success'=> True];
141             header('Content-Type: application/json');
142             echo json_encode($response);
143         }else{
144             $response = ['Success'=> False];
145             header('Content-Type: application/json');
146             echo json_encode($response);
147         }
148
149     0 references | 0 overrides
150     public function changeHotel(){
151         $db = DB::getInstance();
152         $json = file_get_contents('php://input');
153
154         $request = json_decode($json, true);
155         $htl = request['hotel'];
156
157         $fields = array('hNights');
158         $values = array($htl->hNights);
159         $assoc_array = array_combine($fields, $values);
160
161         $where = array('hotelID', $htl->hotelID);
```

```
orderController.php 5 ×

C: > xampp > htdocs > ezgo > orderController.php > PHP > order > orderTour()
  4     class order{
149     public function changeHotel(){
161         if($db->update('hotel', $assoc_array, $where)){
162             $response = ['Success'=> True];
163             header('Content-Type: application/json');
164             echo json_encode($response);
165         }else{
166             $response = ['Success'=> False];
167             header('Content-Type: application/json');
168             echo json_encode($response);
169         }
170     }
171 }
172
173     0 references | 0 overrides
174     public function changeTour(){
175         $db = DB::getInstance();
176         $json = file_get_contents('php://input');
177
178         $request = json_decode($json, true);
179         $trp = request['tour'];
180
181         $fields = array('tpSlot');
182         $values = array($trp->tpSlot);
183         $assoc_array = array_combine($fields, $values);
184
185         $where = array('tourID', $trp->tourID);
186
187         if($db->update('tour_package', $assoc_array, $where)){
188             $response = ['Success'=> True];
189             header('Content-Type: application/json');
190             echo json_encode($response);
191         }else{
192             $response = ['Success'=> False];
193             header('Content-Type: application/json');
194             echo json_encode($response);
195         }
196     }
197     0 references | 0 overrides
198     public function getTicketSingular(){
199         $db = DB::getInstance();
200         $json = file_get_contents('php://input');
201
202         $request = json_decode($json, true);
203         $tid = $request['ticketID'];
204
205         $where = array('ticketID', '=', $tid);
206         $stixs = $db->select('*', 'tickets', $where)->getResults();
207
208         if(count($stixs) > 0){
209             $stix = $stixs[0];
210             $imgPath = $stix->tcImage;
211             $fullPath = "images/".$imgPath;
212             $stix->tcImage = $fullPath;
213
214             $response = ['Success'=> True, 'ticket' => $stix];
215         }
216     }
217 }
```

```
orderController.php 5 ×

C: > xampp > htdocs > ezgo > orderController.php > PHP > order > getTicketSingular()
  4   class order{
197     public function getTicketSingular(){
214       header('Content-Type: application/json');
215       echo json_encode($response);
216     }else{
217       $response = ['Success'=> False];
218       header('Content-Type: application/json');
219       echo json_encode($response);
220     }
221   }

222   0 references | 0 overrides
223   public function getHotelsingular(){
224     $db = DB::getInstance();
225     $json = file_get_contents('php://input');

226     $request = json_decode($json, true);
227     $hid = $request['hotelID'];

228     $where = array('hotelID', '=', $hid);
229     $htls = $db->select('*', 'hotel', $where)->getResults();

230     if(count($htls) > 0){
231       $htl = $htls[0];
232       $imgPath = $htl->hImage;
233       $fullPath = "images/".$imgPath;
234       $htl->hImage = $fullPath;

235       $response = ['Success'=> True, 'hotel' => $htl];
236       header('Content-Type: application/json');
237       echo json_encode($response);
238     }else{
239       $response = ['Success'=> False];
240       header('Content-Type: application/json');
241       echo json_encode($response);
242     }
243   }

244   0 references | 0 overrides
245   public function getTourSingular(){
246     $db = DB::getInstance();
247     $json = file_get_contents('php://input');

248     $request = json_decode($json, true);
249     $trid = $request['tourID'];

250     $where = array('tourID', '=', $trid);
251     $trps = $db->select('*', 'tour_package', $where)->getResults();

252     if(count($trps) > 0){
253       $trp = $trps[0];
254       $imgPath = $trp->tpImage;
255       $fullPath = "images/".$imgPath;
256       $trp->tpImage = $fullPath;

257       $response = ['Success'=> True, 'tour' => $trp];
258       header('Content-Type: application/json');
259     }
260   }

261   0 references | 0 overrides
262   public function getTicketSingular(){
263     $db = DB::getInstance();
264     $json = file_get_contents('php://input');

265     $request = json_decode($json, true);
266     $tid = $request['ticketID'];

267     $where = array('ticketID', '=', $tid);
268     $trps = $db->select('*', 'ticket', $where)->getResults();

269     if(count($trps) > 0){
270       $trp = $trps[0];
271       $imgPath = $trp->tpImage;
272       $fullPath = "images/".$imgPath;
273       $trp->tpImage = $fullPath;

274       $response = ['Success'=> True, 'ticket' => $trp];
275       header('Content-Type: application/json');
276     }
277   }

278   0 references | 0 overrides
279   public function getHotelsingular(){
280     $db = DB::getInstance();
281     $json = file_get_contents('php://input');

282     $request = json_decode($json, true);
283     $hid = $request['hotelID'];

284     $where = array('hotelID', '=', $hid);
285     $htls = $db->select('*', 'hotel', $where)->getResults();

286     if(count($htls) > 0){
287       $htl = $htls[0];
288       $imgPath = $htl->hImage;
289       $fullPath = "images/".$imgPath;
290       $htl->hImage = $fullPath;

291       $response = ['Success'=> True, 'hotel' => $htl];
292       header('Content-Type: application/json');
293       echo json_encode($response);
294     }else{
295       $response = ['Success'=> False];
296       header('Content-Type: application/json');
297       echo json_encode($response);
298     }
299   }

300   0 references | 0 overrides
301   public function getTourSingular(){
302     $db = DB::getInstance();
303     $json = file_get_contents('php://input');

304     $request = json_decode($json, true);
305     $trid = $request['tourID'];

306     $where = array('tourID', '=', $trid);
307     $trps = $db->select('*', 'tour_package', $where)->getResults();

308     if(count($trps) > 0){
309       $trp = $trps[0];
310       $imgPath = $trp->tpImage;
311       $fullPath = "images/".$imgPath;
312       $trp->tpImage = $fullPath;

313       $response = ['Success'=> True, 'tour' => $trp];
314       header('Content-Type: application/json');
315     }
316   }

317   0 references | 0 overrides
318   public function getTicketSingular(){
319     $db = DB::getInstance();
320     $json = file_get_contents('php://input');

321     $request = json_decode($json, true);
322     $tid = $request['ticketID'];

323     $where = array('ticketID', '=', $tid);
324     $trps = $db->select('*', 'ticket', $where)->getResults();

325     if(count($trps) > 0){
326       $trp = $trps[0];
327       $imgPath = $trp->tpImage;
328       $fullPath = "images/".$imgPath;
329       $trp->tpImage = $fullPath;

330       $response = ['Success'=> True, 'ticket' => $trp];
331       header('Content-Type: application/json');
332     }
333   }
```

```
orderController.php 5 X

C: > xampp > htdocs > ezgo > orderController.php > PHP > order > getTourSeveral()

4     class order{
249         public function getTourSeveral(){
267             echo json_encode($response);
268         }else{
269             $response = ['Success'=> False];
270             header('Content-Type: application/json');
271             echo json_encode($response);
272         }
273     }
274

0 references | 0 overrides
275     public function getTicketSeveral(){
276         $db = DB::getInstance();
277         $json = file_get_contents('php://input');

278         $request = json_decode($json, true);
279         $num = $request['number'];

280         $tixs = $db->select('*', 'tickets', null, $num)->getResult();

281         if(count($tixs) > 0){
282             foreach($tixs as $tix){
283                 $imgPath = $tix->tcImage;
284                 $fullPath = "images/".$imgPath;
285                 $tix->tcImage = $fullPath;
286             }

287             $response = ['Success'=> True, 'tickets' => $tixs];
288             header('Content-Type: application/json');
289             echo json_encode($response);
290         }else{
291             $response = ['Success'=> False];
292             header('Content-Type: application/json');
293             echo json_encode($response);
294         }
295     }
296

0 references | 0 overrides
297     public function getHotelSeveral(){
298         $db = DB::getInstance();
299         $json = file_get_contents('php://input');

300         $request = json_decode($json, true);
301         $num = $request['number'];

302         $htls = $db->select('*', 'hotel', null, $num)->getResult();

303         if(count($htls) > 0){
304             foreach($htls as $htl){
305                 $imgPath = $htl->hImage;
306                 $fullPath = "images/".$imgPath;
307                 $htl->hImage = $fullPath;
308             }

309             $response = ['Success'=> True, 'hotels' => $htls];
310             header('Content-Type: application/json');
311             echo json_encode($response);
312         }
313     }
314 }
```

```
orderController.php 5 ×

C: > xampp > htdocs > ezgo > orderController.php > PHP > order > getHotelSeveral()
4     class order{
301         public function getHotelSeveral(){
320             }else{
321                 $response = ['Success'=> False];
322                 header('Content-Type: application/json');
323                 echo json_encode($response);
324             }
325         }
326
327         0 references | 0 overrides
328         public function getTourSeveral(){
329             $db = DB::getInstance();
330             $json = file_get_contents('php://input');

331             $request = json_decode($json, true);
332             $num = $request['number'];

333             $trps = $db->select('*', 'tour_package', null, $num)->getResult();

334             if(count($trps) > 0){
335                 foreach($trps as $trp){
336                     $imgPath = $trp->tpImage;
337                     $fullPath = __images__.'/'.$imgPath;
338                     $trp->tpImage = $fullPath;
339                 }

340                 $response = ['Success'=> True, 'tours' => $trps];
341                 header('Content-Type: application/json');
342                 echo json_encode($response);
343             }else{
344                 $response = ['Success'=> False];
345                 header('Content-Type: application/json');
346                 echo json_encode($response);
347             }
348         }
349
350         0 references | 0 overrides
351         public function getTicketAll(){
352             $db = DB::getInstance();
353             $json = file_get_contents('php://input');

354             $request = json_decode($json, true);
355             $tixs = $db->select('*', 'tickets')->getResult();

356             if(count($tixs) > 0){
357                 foreach($tixs as $tix){
358                     $imgPath = $tix->tcImage;
359                     $fullPath = __images__.'/'.$imgPath;
360                     $tix->tcImage = $fullPath;
361                 }

362                 $response = ['Success'=> True, 'tickets' => $tixs];
363                 header('Content-Type: application/json');
364                 echo json_encode($response);
365             }else{
366                 $response = ['Success'=> False];
367                 header('Content-Type: application/json');
368             }
369         }
370
371         0 references | 0 overrides
372         public function getHotelSeveral(){
373             }else{
374                 $response = ['Success'=> False];
375                 header('Content-Type: application/json');
```

orderController.php 5 X

C: > xampp > htdocs > ezgo > orderController.php > PHP > order > getTicketAll()

```
4     class order{
353         public function getTicketAll(){
373             echo json_encode($response);
374         }
375     }
376
377     0 references | 0 overrides
378     public function getHotelAll(){
379         $db = DB::getInstance();
380         $json = file_get_contents('php://input');
381
382         $request = json_decode($json, true);
383         $htls = $db->select('*', 'hotel')->getResult();
384
385         if(count($htls) > 0){
386             foreach($htls as $htl){
387                 $imgPath = $htl->hImage;
388                 $fullPath = __images__.$imgPath;
389                 $htl->hImage = $fullPath;
390             }
391
392             $response = ['Success'=> True, 'hotels' => $htls];
393             header('Content-Type: application/json');
394             echo json_encode($response);
395         }else{
396             $response = ['Success'=> False];
397             header('Content-Type: application/json');
398             echo json_encode($response);
399         }
400
401     0 references | 0 overrides
402     public function getTourAll(){
403         $db = DB::getInstance();
404         $json = file_get_contents('php://input');
405
406         $request = json_decode($json, true);
407         $trps = $db->select('*', 'tour_package')->getResult();
408
409         if(count($trps) > 0){
410             foreach($trps as $trp){
411                 $imgPath = $trp->tpImage;
412                 $fullPath = __images__.$imgPath;
413                 $trp->tpImage = $fullPath;
414             }
415
416             $response = ['Success'=> True, 'tours' => $trps];
417             header('Content-Type: application/json');
418             echo json_encode($response);
419         }else{
420             $response = ['Success'=> False];
421             header('Content-Type: application/json');
422             echo json_encode($response);
423         }
424     }
425 }
```

```
orderController.php 5  X
C: > xampp > htdocs > ezgo > orderController.php > PHP > order > getTourAll()
4     class order{
425         public function getTicketSpecificType(){
426             $db = DB::getInstance();
427             $json = file_get_contents('php://input');
428
429             $request = json_decode($json, true);
430             $type = $request['type'];
431
432             $where = array('tcType', '=', $type);
433             $tixs = $db->select('*', 'tickets', $where)->getResults();
434
435             if(count($tixs) > 0){
436                 foreach($tixs as $tix){
437                     $imgPath = $tix->tcImage;
438                     $fullPath = "images/".$imgPath;
439                     $tix->tcImage = $fullPath;
440                 }
441
442                 $response = ['Success'=> True, 'tickets' => $tixs];
443                 header('Content-Type: application/json');
444                 echo json_encode($response);
445             }else{
446                 $response = ['Success'=> False];
447                 header('Content-Type: application/json');
448                 echo json_encode($response);
449             }
450         }
451
452         0 references | 0 overrides
453         public function getMixedProd(){
454             $db = DB::getInstance();
455             $json = file_get_contents('php://input');
456
457             $request = json_decode($json, true);
458             $city = $request['city'];
459
460             $tixs = array();
461             $htls = array();
462             $trps = array();
463
464             $where = array('tcFrom', '=', $city);
465             $tixs1 = $db->select('*', 'tickets', $where);
466
467             if(count($tixs1) > 0){
468                 $where = array('tcDestination', '=', $city);
469                 $tixs2 = $db->select('*', 'tickets', $where);
470
471                 if(count($tixs2) > 0){
472                     $tixs = $tixs1 + $tixs2;
473
474                     foreach($tixs as $tix){
475                         $imgPath = $tix->tcImage;
476                         $fullPath = "images/".$imgPath;
477                         $tix->tcImage = $fullPath;
478                     }
479                 }else{
480
481             }
482         }
483     }
484 }
```

```
orderController.php 5 X
C: > xampp > htdocs > ezgo > orderController.php > PHP > order > getMixedProd()
  4   class order{
452     public function getMixedProd(){
479       $response = ['Success'=> False];
480       header('Content-Type: application/json');
481       echo json_encode($response);
482     }
483     }else{
484       $response = ['Success'=> False];
485       header('Content-Type: application/json');
486       echo json_encode($response);
487     }
488
489     $where = array('cityID', '=', $city);
490     $htls = $db->select('*', 'hotel', $where);
491
492     if(count($htls) > 0){
493       $where = array('cityID', '=', $city);
494       $trps = $db->select('*', 'tour_package', $where);
495
496       if(count($trps) > 0){
497         $response = ['Success'=> True, 'tickets' => $tixs, 'hotels' => $htls, 'tours' => $trps];
498         header('Content-Type: application/json');
499         echo json_encode($response);
500       }else{
501         $response = ['Success'=> False];
502         header('Content-Type: application/json');
503         echo json_encode($response);
504       }
505     }else{
506       $response = ['Success'=> False];
507       header('Content-Type: application/json');
508       echo json_encode($response);
509     }
510   }
511
512   0 references | 0 overrides
513   public function searchHotel(){
514     $db = DB::getInstance();
515     $json = file_get_contents('php://input');
516
517     $request = json_decode($json, true);
518     $city = $request['cityID'];
519     $date = $request['hDate'];
520     $room = $request['hRoomType'];
521     $night = $request['hNights'];
522
523     $cols = array('cityID', 'hDate', 'hRoomType', 'hNights');
524     $operator = array('=', '>', '=', '>');
525     $vals = array($city, $date, $room, $night);
526     $where = array($cols, $operator, $vals);
527
528     $htls = $db->select('*', 'hotel', $where)->getResult();
529
530     if(count($htls) > 0){
531       $response = ['Success'=> True, 'Data' => $htls];
532       header('Content-Type: application/json');
533       echo json_encode($response);
534     }
535   }
536 }
```

```
orderController.php 5 ×

C: > xampp > htdocs > ezgo > orderController.php > PHP > order > searchHotel()

 4     class order{
512         public function searchHotel(){
533             }else{
534                 $response = ['Success'=> False];
535                 header('Content-Type: application/json');
536                 echo json_encode($response);
537             }
538         }
539
0 references | 0 overrides
540     public function searchTicket(){
541         $db = DB::getInstance();
542         $json = file_get_contents('php://input');

544         $request = json_decode($json, true);
545         $from = $request['tcFrom'];
546         $to = $request['tcDestination'];
547         $date = $request['tcDate'];
548         $seat = $request['tcSeat'];
549         $type = $request['tcType'];

550
551         $cols = array('tcFrom', 'tcDestination', 'tcDate', 'tcSeat', 'tcType');
552         $operator = array('=', '=', '>=', '>=', '=');
553         $vals = array($from, $to, $date, $seat, $type);
554         $where = array($cols, $operator, $vals);

555
556         $tixs = $db->select('*', 'tickets', $where)->getResult();
557
558         if(count($tixs) > 0){
559             $response = ['Success'=> True, 'Data' => $tixs];
560             header('Content-Type: application/json');
561             echo json_encode($response);
562         }else{
563             $response = ['Success'=> False];
564             header('Content-Type: application/json');
565             echo json_encode($response);
566         }
567     }
568
0 references | 0 overrides
569     public function searchTour(){
570         $db = DB::getInstance();
571         $json = file_get_contents('php://input');

573         $request = json_decode($json, true);
574         $city = $request['cityID'];
575         $date = $request['tpDate'];
576         $slot = $request['tpSlot'];

577
578         $cols = array('cityID', 'tpDate', 'tpSlot');
579         $operator = array('=', '>=', '>=');
580         $vals = array($city, $date, $slot);
581         $where = array($cols, $operator, $vals);

582
583         $trps = $db->select('*', 'tour_package', $where)->getResult();
584
585         if(count($trps) > 0){
```

```
orderController.php 5 X

C:\xampp\htdocs\ezgo\orderController.php > PHP > order > searchTour()

4  class order{
599      public function searchTour(){
586          $response = ['Success'=> True, 'Data' => $trps];
587          header('Content-Type: application/json');
588          echo json_encode($response);
589      }else{
590          $response = ['Success'=> False];
591          header('Content-Type: application/json');
592          echo json_encode($response);
593      }
594  }
595

0 references | 0 overrides
596  public function productSearch(){
597      $db = DB::getInstance();
598
599      $tixs = $db->select('*', 'tickets')->getResultSet();
600
601      if(count($tixs) > 0){
602          foreach($tixs as $tix){
603              $imgPath = $tix->tcImage;
604              $fullPath = "images/".$imgPath;
605              $tix->tcImage = $fullPath;
606          }
607
608          $htls = $db->select('*', 'hotel')->getResultSet();
609
610          if(count($htls) > 0){
611              foreach($htls as $htl){
612                  $imgPath = $htl->hImage;
613                  $fullPath = "images/".$imgPath;
614                  $htl->hImage = $fullPath;
615              }
616
617              $trps = $db->select('*', 'tour_package')->getResultSet();
618
619              if(count($trps) > 0){
620                  foreach($trps as $trp){
621                      $imgPath = $trp->tpImage;
622                      $fullPath = "images/".$imgPath;
623                      $trp->tpImage = $fullPath;
624                  }
625
626                  $cities = $db->select('*', 'city')->getResultSet();
627
628                  if(count($cities) > 0){
629                      $response = ['Success'=> True, 'tickets' => $tixs, 'hotel' => $htls, 'tours' => $trps, 'cities' => $cities];
630                      header('Content-Type: application/json');
631                      echo json_encode($response);
632                  }else{
633                      $response = ['Success'=> False];
634                      header('Content-Type: application/json');
635                      echo json_encode($response);
636                  }
637              }else{
638                  $response = ['Success'=> False];
639                  header('Content-Type: application/json');

```

```

orderController.php 5 ●
C: > xampp > htdocs > ezgo > orderController.php > PHP > order > productSearch()
4   class order{
596     public function productSearch(){
640       echo json_encode($response);
641     }
642     }else{
643       $response = ['Success'=> False];
644       header('Content-Type: application/json');
645       echo json_encode($response);
646     }
647     }else{
648       $response = ['Success'=> False];
649       header('Content-Type: application/json');
650       echo json_encode($response);
651     }
652   }
653
654   0 references | 0 overrides
655   public function homePopular(){
656     $db = DB::getInstance();
657
658     $tixs = $db->select('*', 'tickets', null, 2, 'tcRating', 'DESC')->getResults();
659     $htls = $db->select('*', 'hotel', null, 2, 'hRating', 'DESC')->getResults();
660     $trps = $db->select('*', 'tour_package', null, 2, 'tpRating', 'DESC')->getResults();
661
662     if(count($tixs) && count($htls) && count($trps)){
663       $response = ['Success'=> True, 'Data1' => $tixs, 'Data2' => $htls, 'Data3' => $trps];
664       header('Content-Type: application/json');
665       echo json_encode($response);
666     }else{
667       $response = ['Success'=> False];
668       header('Content-Type: application/json');
669       echo json_encode($response);
670     }
671   }
672 ?>

```

Kode PHP orderController di atas merupakan bagian dari controller yang mengelola pemesanan produk (tiket, hotel, dan paket tour) dalam suatu aplikasi. Controller ini menyediakan berbagai fungsi seperti *orderTicket()*, *orderHotel()*, dan *orderTour()* untuk melakukan pemesanan produk, serta *changeTicket()*, *changeHotel()*, dan *changeTour()* untuk memperbarui informasi pemesanan. Selain itu, terdapat fungsi untuk mendapatkan informasi produk secara tunggal (*getTicketSingular()*, *getHotelSingular()*, *getTourSingular()*), beberapa produk (*getTicketSeveral()*, *getHotelSeveral()*, *getTourSeveral()*), dan semua produk (*getTicketAll()*, *getHotelAll()*, *getTourAll()*). Fungsi pencarian khusus seperti *getTicketSpecificType()*, *searchHotel()*, *searchTicket()*, dan *searchTour()* juga disediakan. Fungsi *homePopular()* mengambil produk populer berdasarkan rating. Semua fungsi ini berinteraksi dengan database

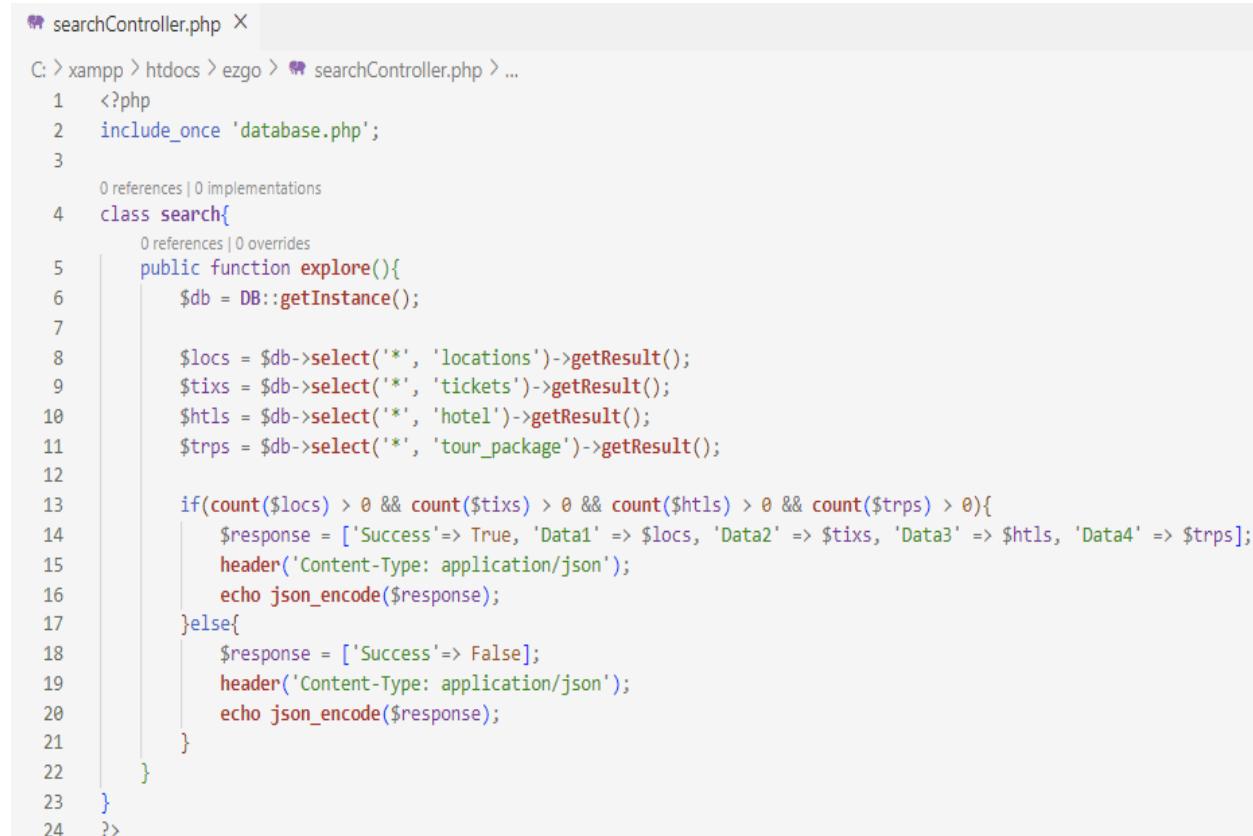
menggunakan kelas *DB* untuk melakukan operasi CRUD dan mengelola transaksi pemesanan. Kode ini mengimplementasikan fitur pemesanan dan manajemen produk yang esensial untuk aplikasi layanan travel berbasis mobile seperti EZGO.

### 3.3.9. *router.php*

```
router.php X
C: > xampp > htdocs > ezgo > router.php
1  <?php
2  if ($_SERVER['REQUEST_METHOD'] === 'POST') {
3      $json = file_get_contents("php://input");
4      $request = json_decode($json, true);
5
6      if (isset($request['controller']) && isset($request['method'])) {
7          $controllerClass = $request['controller'];
8          $controllerMethod = $request['method'];
9
10         $controllerClassName = ($controllerClass) . 'Controller';
11         $controllerFile = $controllerClassName . '.php';
12
13         if (file_exists($controllerFile)) {
14             include $controllerFile;
15             $controller = new $controllerClass();
16
17             if (method_exists($controller, $controllerMethod)) {
18                 $controller->$controllerMethod();
19             } else {
20                 $response = ['Success' => False, 'Message' => 1];
21                 header('Content-Type: application/json');
22                 echo json_encode($response);
23             }
24         } else {
25             $response = ['Success' => False, 'Message' => 2];
26             header('Content-Type: application/json');
27             echo json_encode($response);
28         }
29     } else {
30         $response = ['Success' => False, 'Message' => 3];
31         header('Content-Type: application/json');
32         echo json_encode($response);
33     }
34 } else {
35     $response = ['Success' => False, 'Message' => 4];
36     header('Content-Type: application/json');
37     echo json_encode($response);
38 }
39 ?>
```

Kode PHP router di atas merupakan router yang berfungsi sebagai pengatur jalur untuk permintaan *POST* dalam aplikasi. Router ini membaca *input JSON* dari permintaan HTTP *POST* dan mendekodekannya untuk mendapatkan nama controller dan metode yang akan dijalankan. Jika controller dan metode yang diminta ada, router akan menyertakan file controller yang sesuai, membuat instance dari controller tersebut, dan memanggil metode yang diminta. Jika controller atau metode tidak ditemukan, atau jika ada kesalahan dalam permintaan, router akan mengembalikan *response JSON* dengan pesan kesalahan yang sesuai. Kode ini mengimplementasikan fitur routing dinamis, yang memungkinkan pengelolaan dan eksekusi permintaan ke berbagai controller dan metode berdasarkan input dari pengguna, sehingga memudahkan pengembangan aplikasi yang modular dan terstruktur.

### 3.3.10. searchController.php



```
C:\xampp\htdocs\ezgo> searchController.php ...
1 <?php
2 include_once 'database.php';
3
4 class search{
5     0 references | 0 implementations
6     public function explore(){
7         $db = DB::getInstance();
8
9         $locs = $db->select('*', 'locations')->getResult();
10        $tixs = $db->select('*', 'tickets')->getResult();
11        $htls = $db->select('*', 'hotel')->getResult();
12        $trps = $db->select('*', 'tour_package')->getResult();
13
14        if(count($locs) > 0 && count($tixs) > 0 && count($htls) > 0 && count($trps) > 0){
15            $response = ['Success'=> True, 'Data1' => $locs, 'Data2' => $tixs, 'Data3' => $htls, 'Data4' => $trps];
16            header('Content-Type: application/json');
17            echo json_encode($response);
18        }else{
19            $response = ['Success'=> False];
20            header('Content-Type: application/json');
21            echo json_encode($response);
22        }
23    }
24 ?>
```

Kode PHP *searchController* di atas merupakan bagian dari controller *search* yang menyediakan fungsi eksplorasi dalam suatu aplikasi. Fungsi *explore()* digunakan untuk mengambil data dari beberapa tabel dalam database *locations*, *tickets*, *hotel*, dan *tour\_package*.

Setelah data diambil, fungsi ini memeriksa apakah setiap tabel memiliki setidaknya satu entri. Jika semua tabel mengandung data, fungsi ini mengembalikan *respons JSON* yang berisi data dari setiap tabel tersebut. Jika salah satu tabel tidak mengandung data, fungsi ini mengembalikan *respons JSON* dengan status *Success* yang bernilai *False*. Kode ini mengimplementasikan fitur eksplorasi data yang memungkinkan pengguna untuk mendapatkan informasi lengkap tentang berbagai kategori produk seperti lokasi, tiket, hotel, dan paket tour dalam aplikasi.

### 3.3.11. transactionController.php

```
transactionController.php X
C: > xampp > htdocs > ezgo > transactionController.php > ...
1  <?php
2  include_once 'database.php';
3
4  class transaction{
5      0 references | 0 implementations
6      0 references | 0 overrides
7      public function transAll(){
8          $db = DB::getInstance();
9          $json = file_get_contents('php://input');
10
11         $request = json_decode($json, true);
12         $uid = $request['userID'];
13
14         $where = array('userID', '=', $uid);
15         $trids = $db->select('productID', 'transaction_details', $where)->getResultSet();
16         $tixs = $db->select('*', 'tickets')->getResultSet();
17         $htls = $db->select('*', 'hotel')->getResultSet();
18         $trps = $db->select('*', 'tour_package')->getResultSet();
19
20         if(count($trids) > 0 && count($tixs) > 0 && count($htls) > 0 && count($trps) > 0){
21             $id = array();
22
23             foreach($trids as $trid){
24                 $id[] = $trid->productID;
25             }
26
27             $response = ['Success' => True, 'Data1' => $id, 'Data2' => $tixs, 'Data3' => $htls, 'Data4' => $trps];
28             header('Content-Type: application/json');
29             echo json_encode($response);
30         }else{
31             $response = ['Success' => False];
32             header('Content-Type: application/json');
33             echo json_encode($response);
34         }
35     }
36  ?>
```

Kode PHP `transactionController` di atas merupakan bagian dari controller `transaction` yang mengelola transaksi pengguna dalam suatu aplikasi. Fungsi `transAll()` digunakan untuk mengambil semua transaksi yang terkait dengan `userID` tertentu. Fungsi ini membaca permintaan HTTP `POST` dalam format `JSON` untuk mendapatkan `userID`, kemudian mengambil data `productID` dari tabel `transaction_details` yang sesuai dengan `userID` tersebut. Selain itu, fungsi ini juga mengambil semua data dari tabel `tickets`, `hotel`, dan `tour_package`. Jika data transaksi dan data dari tabel-tabel tersebut ditemukan, fungsi ini mengembalikan `response JSON` yang berisi `productID` dari transaksi pengguna dan data lengkap dari tiket, hotel, dan paket tour. Jika tidak ditemukan, fungsi ini mengembalikan `response JSON` dengan status `Success` yang bernilai `False`. Kode ini mengimplementasikan fitur manajemen transaksi, memungkinkan pengguna untuk melihat semua transaksi terkait produk yang mereka pesan dalam aplikasi.

## **BAB IV**

### **PENUTUP**

#### **4.1. Catatan Aspek Penilaian**

Berdasarkan berbagai kriteria penilaian yang telah ditetapkan pada soal UAS Mobile Application Development kali ini, maka pada sub bab ini kami akan menjelaskan aspek-aspek penilaian yang menjadi fokus utama dalam pengembangan aplikasi EZGO. Setiap aspek penilaian mencakup elemen-elemen kunci yang dinilai untuk memastikan bahwa aplikasi ini memenuhi standar kualitas yang diharapkan. Beberapa poin penting yang berhasil diimplementasikan adalah:

- **Database Design dan Real-time Integration**

Aplikasi EZGO berhasil mengimplementasikan desain database yang optimal dan efektif untuk menyimpan dan mengelola data. Kami menggunakan XAMPP sebagai platform server lokal untuk mendukung pengembangan dan pengujian database. Integrasi database secara real-time memastikan bahwa semua aktivitas pengguna, mulai dari pendaftaran hingga transaksi pembayaran, tersimpan dan diperbarui secara mutakhir. Setiap kali ada data yang ditambahkan melalui aplikasi mobile, seperti pendaftaran pengguna baru atau pemesanan tiket/ hotel/ paket tour, data tersebut juga secara otomatis akan diperbarui di database XAMPP. Hal ini meningkatkan keamanan dan keandalan aplikasi karena data tersimpan dengan konsisten dan akurat di server.

- **Web View**

Implementasi WebView pada aplikasi EZGO digunakan untuk meningkatkan pengalaman pengguna dalam mengeksplorasi destinasi wisata. Pada menu detail eksplorasi destinasi wisata, kami menyediakan fitur untuk melihat video gambaran kondisi destinasi wisata melalui YouTube. Setiap destinasi wisata memiliki link video YouTube yang berbeda, disesuaikan dengan destinasi terkait. Dengan WebView ini, pengguna dapat menonton video yang relevan dengan destinasi wisata yang mereka minati, memberikan mereka gambaran visual yang lebih kaya tentang tempat wisata tersebut. Fitur ini memastikan bahwa pengguna mendapatkan informasi tambahan yang berguna dan relevan, sehingga mereka dapat membuat keputusan yang lebih baik

mengenai destinasi yang akan dikunjungi. Integrasi WebView ini juga memastikan bahwa semua konten multimedia dapat diakses dengan lancar dan tanpa gangguan.

- **ViewPager**

Implementasi ViewPager pada aplikasi EZGO digunakan untuk meningkatkan pengalaman pengguna dalam menjelajahi berbagai halaman awal aplikasi. Kami menerapkan ViewPager pada menu splash screen aplikasi, yang muncul sebelum pengguna melakukan registrasi atau login akun. Ketika aplikasi pertama kali dimulai, pengguna akan melihat dua halaman splash screen yang berisi gambar dan kata-kata pembuka aplikasi EZGO, diikuti oleh satu halaman login/registrasi. Ketiga halaman ini menggunakan konsep ViewPager, sehingga pengguna dapat berpindah antar halaman dengan menggeser (slide) ke samping kiri atau kanan. Dengan ViewPager, transisi antar halaman menjadi lebih halus dan intuitif. Integrasi ViewPager ini memastikan bahwa pengguna baru mendapatkan pengenalan yang jelas terhadap aplikasi sebelum mulai menggunakan layanan-layanan yang tersedia.

- **SQLite**

Implementasi SQLite pada aplikasi EZGO terlihat pada kelas *internalDB*. Kelas ini mengelola database lokal untuk menyimpan informasi pengguna. Kelas *internalDB* memperluas *SQLiteOpenHelper* dan mengimplementasikan metode *onCreate()* untuk membuat tabel *user* yang menyimpan informasi seperti ID, nama, alamat, telepon, email, tanggal lahir, dan foto profil. Selain itu, kelas ini menyediakan berbagai metode untuk operasi CRUD (Create, Read, Update, Delete) pada data pengguna. Metode *createUser()* digunakan untuk menambahkan pengguna baru ke database, *getUser()* untuk mengambil informasi pengguna, *updateUser()* untuk memperbarui data pengguna, dan *deleteUser()* untuk menghapus data pengguna dari database. Implementasi ini memungkinkan aplikasi untuk mengelola data pengguna secara lokal dengan efisien, memastikan bahwa informasi penting tetap tersedia bahkan tanpa koneksi internet. Sebagai contoh, pada kelas *LandingActivity*, data pengguna diambil dari SQLite melalui *internalDB* dan disimpan kembali ke *SharedPreferences* jika ada, sehingga pengguna dapat langsung masuk ke aplikasi tanpa harus login ulang. Pendekatan ini memberikan efisiensi dan

meningkatkan pengalaman pengguna dengan menyediakan akses cepat dan andal ke informasi pengguna yang tersimpan secara lokal.

- **PHP script as API (database connection, get data, add data, insert data, edit data, delete data)**

Aplikasi EZGO menggunakan skrip PHP sebagai API untuk menghubungkan aplikasi mobile dengan database di XAMPP, yang memungkinkan operasi CRUD (Create, Read, Update, Delete) dilakukan dengan lancar. Implementasi ini mencakup beberapa aspek penting, seperti mengelola koneksi database, memperoleh data, menambah data, memasukkan data, mengedit data, dan menghapus data. Misalnya, ketika pengguna mendaftar akun baru melalui aplikasi mobile, skrip PHP akan menerima permintaan tersebut, memproses data yang diberikan (seperti nama lengkap, email, username, dan password), dan kemudian memperbarui database di XAMPP secara real-time menggunakan MySQL.

Fungsi-fungsi dalam skrip PHP, seperti dalam *accountController.php*, mengelola informasi akun pengguna dengan menyediakan metode untuk menampilkan, memperbarui, dan menghapus akun berdasarkan userID. Selain itu, *cityController.php* dan *exploreController.php* digunakan untuk mengelola data kota dan lokasi wisata, sementara *locationController.php* menangani data lokasi populer dan pembaruan jumlah likes. Proses parsing JSON dari API PHP ke aplikasi Android memastikan bahwa data yang diperoleh atau diubah dapat ditampilkan atau disimpan dengan tepat di aplikasi mobile. *router.php* mengatur jalur permintaan POST, memastikan bahwa permintaan yang datang ditangani oleh controller yang tepat, dan *transactionController.php* mengelola transaksi pengguna dengan mengambil semua data transaksi terkait produk yang dipesan. Implementasi ini memastikan bahwa setiap perubahan atau penambahan data di aplikasi mobile secara otomatis diperbarui di database XAMPP, sehingga data selalu konsisten dan akurat, meningkatkan keandalan aplikasi EZGO.

- **Parsing JSON data to Android for database connections, get data, add data, insert data, edit data, and delete (CRUD) data.**

Penerapan parsing data JSON untuk koneksi database, pengambilan data, penambahan data, penyisipan data, pengeditan data, dan penghapusan data (CRUD) telah

suskes dilakukan dengan menggunakan kombinasi library *Volley* dan *Gson*. *Volley* digunakan untuk mengelola permintaan HTTP ke server, sementara *Gson* digunakan untuk mengonversi data JSON ke objek Java dan sebaliknya. Dalam proses pengambilan data, aplikasi mengirimkan permintaan HTTP *POST* atau *GET* ke server, menerima data dalam format JSON, dan mengonversinya menjadi objek Java menggunakan *Gson*. Penambahan data dilakukan dalam aktivitas seperti *SignupActivity*, di mana data pengguna baru dikirimkan ke server dalam format JSON melalui permintaan HTTP *POST* menggunakan *Volley*. Penyisipan data ke database lokal dikelola oleh kelas *internalDB*, yang menangani operasi CRUD menggunakan *SQLite*. Pembaruan data, seperti dalam *AccountInformationActivity*, dilakukan dengan mengirimkan data yang diperbarui ke server dalam format JSON, kemudian menyimpannya kembali di database lokal jika diperlukan. Penghapusan data ditangani dalam *ProfileFragment*, di mana permintaan untuk menghapus akun pengguna dikirim ke server, dan data pengguna dihapus dari *SQLite* dan *SharedPreferences*. Implementasi ini memastikan bahwa aplikasi dapat mengelola operasi CRUD dengan optimal, mempertukarkan data dengan server dalam format JSON, dan menyimpan data secara lokal untuk memastikan ketersediaan meskipun tanpa koneksi internet.

- **Penerapan CRUD**

- **Create**

1. *AccountInformationActivity (Android)* → Saat pengguna mendaftar, informasi akun baru dikirim ke server melalui permintaan HTTP *POST* yang dikirimkan oleh *Volley*, dan kemudian disimpan di database.
2. *SignupActivity (Android)* → Saat pengguna membuat akun baru, data pengguna dikirim ke server dan disimpan dalam database melalui endpoint *createAccount* di *loginController.php*.
3. *orderController.php (PHP)* → Metode seperti *orderTicket()*, *orderHotel()*, dan *orderTour()* digunakan untuk membuat pesanan baru berdasarkan informasi yang dikirim dari aplikasi mobile.

- **Read**

1. *HomeFragment (Android)* → Data lokasi populer dan produk populer diambil dari server menggunakan Volley dan ditampilkan dalam RecyclerView.
2. *ExploreFragment (Android)* → Data lokasi diambil dari server menggunakan Volley, kemudian dikonversi dari JSON menjadi objek location dan ditampilkan dalam RecyclerView.
3. *AccountInformationActivity (Android)* → Saat aktivitas dibuat, informasi akun pengguna diambil dari SharedPreferences dan ditampilkan di UI.
4. *orderController.php (PHP)* → Fungsi getTicketAll(), getHotelAll(), dan getTourAll() digunakan untuk mengambil semua data tiket, hotel, dan paket tour dari database untuk ditampilkan dalam aplikasi.

- **Update**

1. *AccountInformationActivity (Android)* → Pengguna dapat memperbarui informasi akun mereka. Data yang diperbarui dikirim ke server melalui permintaan HTTP POST, disimpan dalam database, dan kemudian diambil dan ditampilkan kembali di UI.
2. *ChangePassActivity (Android)* → Pengguna dapat memperbarui kata sandi mereka dengan mengirimkan data baru ke server melalui permintaan HTTP POST, yang kemudian diperbarui di database.
3. *ProfileFragment (Android)* → Pengguna dapat memperbarui informasi profil mereka. Data baru dikirim ke server dan diupdate dalam database.
4. *accountController.php (PHP)* → Fungsi updateAccount() digunakan untuk memperbarui informasi akun pengguna, termasuk gambar profil, nama, alamat, telepon, email, dan tanggal lahir.

- **Delete**

1. *ProfileFragment (Android)* → Pengguna dapat menghapus akun mereka. Permintaan untuk menghapus akun dikirim ke server, dan data pengguna dihapus dari database dan SharedPreferences.
2. *accountController.php (PHP)* → Fungsi delete() digunakan untuk menghapus akun pengguna berdasarkan userID yang diterima dari permintaan HTTP.

3. *orderController.php (PHP)* → Fungsi seperti `changeTicket()`, `changeHotel()`, dan `changeTour()` dapat digunakan untuk menghapus atau memperbarui informasi pemesanan yang ada.

- **Maps API**

Pada kode *ExploreDetailActivity* dalam aplikasi Android, terdapat penerapan Maps API untuk menampilkan lokasi pada peta interaktif. Saat aktivitas ini dibuat, elemen UI seperti `ImageView`, `TextView`, `ToggleButton`, dan `MaterialButton` diinisialisasi. Aktivitas ini menerima objek `location` dari intent dan menampilkan detailnya seperti nama, kota, deskripsi, dan gambar menggunakan Picasso. Informasi kota diambil dari server menggunakan Volley dan ditampilkan. Lokasi juga ditampilkan di peta menggunakan Google Maps API. Penggunaan Maps API memungkinkan pengguna untuk melihat lokasi eksplorasi secara visual dan interaktif, dengan peta yang menampilkan detail posisi geografis tempat tersebut. Selain itu, tombol suka (`btnLike`) memungkinkan pengguna untuk menambah atau mengurangi jumlah suka, yang kemudian diperbarui di server, serta tombol tonton (`btnWatch`) membuka `WebViewActivity` untuk melihat link terkait lokasi. Integrasi ini memberikan informasi lokasi yang lengkap dan mudah dipahami.

## 4.2. Kesimpulan

Aplikasi EZGO telah berhasil dikembangkan dengan tujuan utama untuk memudahkan masyarakat dalam mengakses informasi perjalanan dan pemesanan layanan wisata melalui platform mobile yang cepat dan mudah digunakan. Aplikasi ini menyediakan fitur pencarian dan pemesanan tiket, hotel, dan paket tour yang terintegrasi, memungkinkan pengguna untuk merencanakan perjalanan mereka dengan lebih optimal. Dengan informasi harga, visual yang beragam, dan ketersediaan banyak layanan yang transparan dan real-time, EZGO membantu mengurangi ketidakpastian yang sering dihadapi pengguna. Pengguna dapat dengan mudah melakukan transaksi pembayaran online berkat implementasi teknologi terbaru, yang mendorong inovasi dalam industri travel dan memenuhi tuntutan konsumen modern. Selain itu, fitur seperti riwayat pembelian dan pengelolaan profil pengguna memastikan bahwa semua informasi dan

aktivitas pengguna tersimpan dan diperbarui secara akurat, meningkatkan keamanan dan kenyamanan pengguna selama perjalanan.

Dengan memanfaatkan data untuk personalisasi, EZGO mampu merekomendasikan destinasi wisata yang populer dan relevan, memberikan pengalaman yang lebih sesuai dengan preferensi pengguna. Fitur pencarian yang mendukung kata kunci spesifik memudahkan pengguna menemukan destinasi yang diinginkan dengan cepat. Integrasi database secara real-time memastikan bahwa setiap perubahan atau penambahan data diperbarui secara otomatis, meningkatkan keandalan aplikasi. Implementasi WebView dan Maps API memberikan informasi visual yang kaya dan interaktif tentang destinasi wisata, membantu pengguna membuat keputusan yang lebih baik. Secara keseluruhan, EZGO berhasil mengatasi keterbatasan aplikasi travel saat ini dengan menyediakan pembaruan dan akses mudah ke pemberitahuan perjalanan, serta pengalaman pemesanan yang intuitif dan aman, menjadikannya solusi yang inovatif dalam industri travel yang dinamis.

### **4.3. Saran**

Untuk meningkatkan aplikasi EZGO, disarankan untuk menambahkan fitur ulasan dan penilaian dari pengguna untuk tiket, hotel, dan paket tour, sehingga memberikan pandangan yang lebih komprehensif bagi calon pengguna. Integrasi dengan platform media sosial dapat memudahkan pengguna dalam berbagi pengalaman perjalanan mereka dan menarik lebih banyak pengguna baru. Mengimplementasikan fitur notifikasi push untuk pembaruan harga atau ketersediaan tiket, serta penawaran khusus juga dapat meningkatkan keterlibatan pengguna. Selain itu, memperluas opsi pembayaran dengan berbagai metode pembayaran digital yang lebih beragam akan memberikan fleksibilitas lebih bagi pengguna dalam melakukan transaksi. Fitur blog dan chat interaktif juga sangat direkomendasikan. Blog dapat digunakan untuk membagikan cerita pengalaman perjalanan, tips perjalanan, dan ulasan destinasi. Sementara chat interaktif memungkinkan pengguna untuk berkomunikasi langsung dengan agen perjalanan atau sesama pengguna, memungkinkan diskusi tentang rencana perjalanan, pertanyaan, dan saran secara real-time. Dengan fitur-fitur ini, aplikasi EZGO tidak hanya menjadi platform pemesanan, tetapi juga komunitas bagi para traveler untuk berbagi pengalaman dan mendapatkan informasi yang berguna, meningkatkan engagement dan loyalitas pengguna.

## LAMPIRAN

Dikarenakan ukuran file aplikasi dan video yang terlalu besar untuk diunggah ke e-learning, kami melampirkan link Google Drive untuk aplikasi dan videonya. Berikut link Google Drive untuk video dan aplikasi:

[https://drive.google.com/drive/folders/1dxs56Qyrczu7k\\_RNo7hhFhcCOlTzY-gn?usp=sharing](https://drive.google.com/drive/folders/1dxs56Qyrczu7k_RNo7hhFhcCOlTzY-gn?usp=sharing)

Jika mengalami kesulitan dalam membuka atau terdapat kesalahan pada file, mohon segera hubungi kami melalui email atau LINE pribadi Karin.

- Email: [karin.eldora@student.umn.ac.id](mailto:karin.eldora@student.umn.ac.id)
- ID LINE: karineldora

## PERAN MASING-MASING ANGGOTA

Berikut merupakan rincian pembagian tugas kelompok EZGO dalam penyelesaian proyek akhir sebagai Ujian Akhir Semester mata kuliah Mobile Application Development (IS534-A & IS 534-AL). Setiap anggota kelompok memiliki peran dan tanggung jawab masing-masing. Dengan pembagian tugas yang jelas, kami dapat memastikan hasil yang optimal dan sesuai dengan tujuan proyek. Berikut adalah peran dari masing-masing anggota kelompok dalam pembuatan aplikasi EZGO:

1. **Karin Eldora (NIM: 68097)** → Membuat proposal, Database Design, Coding BackEnd, Design Interface Aplikasi, dan Coding pada Android Studio.
2. **Kenny Lawson (NIM: 81065)** → Membantu Pembuatan PPT, Proses Bisnis, dan Membantu Design Interface Aplikasi.
3. **Jantzen Fernandes (NIM: 74907)** → Membuat PPT, Membantu Database Design, Membantu Coding pada Android Studio.