Chapter 4:

Coding

4.1Algorithm:

1. Login Module:

- Prompt the user to enter their username and password.
- Verify the entered credentials against the database.
- If the credentials are valid, authenticate the user and grant access to the application.
- If the credentials are invalid, display an error message and prompt the user to re-enter their credentials.
- Allow users to reset their password if they have forgotten it.

2. Home Module:

- Display a dashboard with various options such as viewing services, latest offers, and packages.
- Retrieve and display the latest offers and promotions from the database.
- Retrieve and display a list of available services for users to browse.
- Provide links or buttons for users to navigate to other modules such as booking, location, user profile, etc.

3. Booking Module:

- Allow users to select the type of service they require (e.g., maintenance, repair, detailing).
- Provide options for users to select the date and time for their appointment.
- Allow users to choose the specific car for which they are booking the service.
- Validate the selected options and confirm the booking request.

- Store the booking details in the database and send a confirmation notification to the user.

4. Location Module:
    - Retrieve the user's current location using GPS or device sensors.
    - Allow users to share their location with the service provider for emergency assistance or service appointments.
    - Provide options for users to input additional location details or instructions if needed.
    - Send the location information to the backend server for processing and storage.

5. User Profile Module:
    - Display the user's profile information such as name, contact details, and address.
    - Allow users to edit their profile information if necessary.
    - Provide options for users to add or remove cars from their profile.
    - Allow users to view detailed information about each car in their profile, including make, model, year, etc.
    - Store any changes made to the user profile in the database.

6. Booking Request Module:
    - Retrieve a list of pending booking requests from the database.
    - Display the booking requests to the service provider or administrator.
    - Allow service providers to accept or reject booking requests based on availability and capacity.
    - Update the status of the booking request in the database accordingly.

7. View Request Module:

- Retrieve a list of previous and upcoming service requests from the database.
- Display detailed information about each service request, including date, time, type of service, car details, etc.
- Allow users to view the status of each service request (e.g., pending, confirmed, completed).
- Provide options for users to cancel or reschedule upcoming service requests if needed.

8. Manage Mechanic Module:

- Retrieve a list of available mechanics or service providers from the database.
- Allow administrators to add new mechanics or remove existing ones from the system.
- Provide options for administrators to assign specific tasks or appointments to individual mechanics.
- Allow administrators to track the performance and availability of each mechanic over time.

9. Logout Module:

- Provide users with an option to log out of their account.
- Clear any active sessions and revoke access to the application.
- Redirect users to the login screen or homepage after successful logout.

4.2 Code Snippet:

1. Home page:

```
package com.example.carproject

import android.annotation.SuppressLint
import android.app.Dialog
import android.content.ClipData
import android.content.ClipboardManager
import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.*
import com.example.carproject.Adapter.CarTypeAdapter
import com.example.carproject.Utils.InternetConnectivityUtil
import com.example.carproject.databinding.FragmentHomeBinding
import com.google.android.gms.ads.AdRequest
import com.google.android.gms.ads.MobileAds
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

class Home : Fragment() {
    private var selectedOilType: String = ""
    val user = FirebaseAuth.getInstance().currentUser
    //private lateinit var supportFragmentManager: FragmentManager
    private val databaseReference =

FirebaseDatabase.getInstance().getReference("Customer").child(user!
!.uid)

    private lateinit var fragmentHomeBinding: FragmentHomeBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        fragmentHomeBinding = FragmentHomeBinding.inflate(inflater,
container, false)
        val view = fragmentHomeBinding.root

        // CHECK INTERNET
        if
(!InternetConnectivityUtil.isInternetAvailable(requireContext())
        ) {
            if (container != null) {
                InternetConnectivityUtil.showInternetSnackbar(container,
requireContext())
            }
        }

        // OIL CHANGE
        fragmentHomeBinding.clOilchange.setOnClickListener  {
         asd("oil",com.example.carproject.R.array.oil_types)
        }



        // Replacebattery
        fragmentHomeBinding.clReplacebattery.setOnClickListener {
        asd("awsd",com.example.carproject.R.array.battery_types)
        }
        // DISPLAY ADS
        MobileAds.initialize(requireActivity())  {}

        var mAdView = fragmentHomeBinding.adView
        val adRequest = AdRequest.Builder().build()
        mAdView.loadAd(adRequest)

        //DISPLAY CURRENT USER'S NAME
        databaseReference.addValueEventListener(object                        :
ValueEventListener {
            @SuppressLint("SetTextI18n")
            override fun onDataChange(dataSnapshot: DataSnapshot) {
                if (dataSnapshot.exists()) {
                    val name = dataSnapshot.child("name").value.toString()
                    val                    usernumber                    =
dataSnapshot.child("mobile").value.toString()
                    fragmentHomeBinding.tvhii.text = "Hii, $name"
                    fragmentHomeBinding.tvusernumber.text          =         "
$usernumber"
                }
            }

            override fun onCancelled(databaseError: DatabaseError) {
                // Handle errors
            }
        })

        // GO TO EDIT PROFILE
        fragmentHomeBinding.imgProfile.setOnClickListener   {
           val intent = Intent(requireContext(), EditProfile::class.java)
           startActivity(intent)
        }

        // GO TO BOOK SERVICE
        fragmentHomeBinding.clScheduleTime.setOnClickListener  {
           (activity as Dashboard).goToBooking()
        }

        // OFFERS - CarEngine25
        fragmentHomeBinding.couponCarEngine25.setOnClickListener
{
           // Create a custom dialog box
           val dialog = Dialog(requireContext())
           dialog.setContentView(R.layout.dialog_carengine25)

           // Set the coupon text
           val                              couponText                              =
dialog.findViewById<TextView>(R.id.coupon_text)
           //couponText.text = "CARENGINE25"

           // Set the icon click listener to copy the coupon text to clipboard
           val                              copyIcon                              =
dialog.findViewById<ImageView>(R.id.copy_icon)
            copyIcon.setOnClickListener {
               // Get the clipboard manager
              val                              clipboard                              =
requireContext().getSystemService(Context.CLIPBOARD_SERVICE
) as ClipboardManager
```

```
                // Create a clip with the coupon text
                val clip = ClipData.newPlainText("Coupon",
couponText.text)

                // Set the clip on the clipboard
                clipboard.setPrimaryClip(clip)

                // Show a toast message indicating that the coupon has been
copied
                Toast.makeText(requireContext(), "Coupon copied",
Toast.LENGTH_SHORT).show()

                // Dismiss the dialog box
                dialog.dismiss()
            }

            // Show the dialog box
            dialog.show()
        }

        // OFFERS - CarOiling15
        fragmentHomeBinding.couponCarOiling15.setOnClickListener {
            // Create a custom dialog box
            val dialog = Dialog(requireContext())
            dialog.setContentView(R.layout.dialog_caroiling15)

            // Set the coupon text
            val couponText =
dialog.findViewById<TextView>(R.id.coupon_text)

            // Set the icon click listener to copy the coupon text to clipboard
            val copyIcon =
dialog.findViewById<ImageView>(R.id.copy_icon)
             copyIcon.setOnClickListener {
                // Get the clipboard manager
                val clipboard =
requireContext().getSystemService(Context.CLIPBOARD_SERVICE
) as ClipboardManager

                // Create a clip with the coupon text
                val clip = ClipData.newPlainText("Coupon",
couponText.text)

                // Set the clip on the clipboard
                clipboard.setPrimaryClip(clip)

                // Show a toast message indicating that the coupon has been
copied
                Toast.makeText(requireContext(), "Coupon copied",
Toast.LENGTH_SHORT).show()

                // Dismiss the dialog box
                dialog.dismiss()
            }

            // Show the dialog box
            dialog.show()
        }

        // OFFERS - CarTyre25
        fragmentHomeBinding.couponCarTyre25.setOnClickListener {
            // Create a custom dialog box
            val dialog = Dialog(requireContext())
            dialog.setContentView(R.layout.dialog_cartyre25)
            // Set the coupon text
            val couponText =
dialog.findViewById<TextView>(R.id.coupon_text)

            // Set the icon click listener to copy the coupon text to clipboard
            val copyIcon =
dialog.findViewById<ImageView>(R.id.copy_icon)
             copyIcon.setOnClickListener {
                // Get the clipboard manager
                val clipboard =
requireContext().getSystemService(Context.CLIPBOARD_SERVICE
) as ClipboardManager

                // Create a clip with the coupon text
                val clip = ClipData.newPlainText("Coupon",
couponText.text)

                // Set the clip on the clipboard
                clipboard.setPrimaryClip(clip)

                // Show a toast message indicating that the coupon has been
copied
                Toast.makeText(requireContext(), "Coupon copied",
Toast.LENGTH_SHORT).show()

                // Dismiss the dialog box
                dialog.dismiss()
            }

            // Show the dialog box
            dialog.show()
        }
```

```
        // OFFERS - CarWash1500

fragmentHomeBinding.couponCarWash1500.setOnClickListener  {
            // Create a custom dialog box
            val dialog = Dialog(requireContext())
            dialog.setContentView(R.layout.dialog_carwash1500)

            // Set the coupon text
            val                          couponText                              =
dialog.findViewById<TextView>(R.id.coupon_text)

            // Set the icon click listener to copy the coupon text to clipboard
            val                          copyIcon                                =
dialog.findViewById<ImageView>(R.id.copy_icon)
             copyIcon.setOnClickListener {
                 // Get the clipboard manager
                val                          clipboard                            =
requireContext().getSystemService(Context.CLIPBOARD_SERVICE
) as ClipboardManager

                // Create a clip with the coupon text
                val        clip        =        ClipData.newPlainText("Coupon",
couponText.text)

                // Set the clip on the clipboard
                clipboard.setPrimaryClip(clip)

                // Show a toast message indicating that the coupon has been
copied
                Toast.makeText(requireContext(),       "Coupon       copied",
Toast.LENGTH_SHORT).show()
```

```
            // Dismiss the dialog box
            dialog.dismiss()
        }

        // Show the dialog box
        dialog.show()
    }

    // BASIC PACKAGE
    fragmentHomeBinding.clBasicpkcg.setOnClickListener {
        val dialog = Dialog(requireContext())
        dialog.setContentView(R.layout.popup_basicservice)
        dialog.setCanceledOnTouchOutside(false)
        val                    closeButton                    =
dialog.findViewById<Button>(R.id.btn_closebasic)
        closeButton.setOnClickListener {
            dialog.dismiss()
        }
        dialog.show()
    }

    // STANDARD PACKAGE
    fragmentHomeBinding.clStandardckg.setOnClickListener {
        val dialog = Dialog(requireContext())
        dialog.setContentView(R.layout.popup_standardservice)
        dialog.setCanceledOnTouchOutside(false)
        val                    closeButton                    =
dialog.findViewById<Button>(R.id.btn_closestandard)
        closeButton.setOnClickListener {
            dialog.dismiss()
        }
        dialog.show()
    }
    // PREMIUM PACKAGE
    fragmentHomeBinding.clPremiumpckg.setOnClickListener {
        val dialog = Dialog(requireContext())
        dialog.setContentView(R.layout.popup_premiumservice)
        dialog.setCanceledOnTouchOutside(false)
        val closeButton =
dialog.findViewById<Button>(R.id.btn_closepremium)
        closeButton.setOnClickListener {
            dialog.dismiss()
        }
        dialog.show()
    }
    return view
}

private fun asd(miniservicename: String, servicetype: Int) {
    val dialog = Dialog(requireContext())
    dialog.setContentView(R.layout.activity_mini_service)


    val spinner =
dialog.findViewById<Spinner>(R.id.spinner_OilType)
    val oilTypes = resources.getStringArray(servicetype)
    val adapter_cartype = CarTypeAdapter(requireContext(),
oilTypes)
    spinner.setAdapter(adapter_cartype);
    spinner.onItemSelectedListener =
        object : AdapterView.OnItemSelectedListener {
            override fun onItemSelected(
                parent: AdapterView<*>?,
                view: View?,
                position: Int,
                id: Long,
            ) {
                if (position > 0) {
                    selectedOilType = oilTypes[position]
                    Toast.makeText(
                        requireContext(),
                        "$miniservicename: $selectedOilType",
                        Toast.LENGTH_SHORT
                    ).show()
                }
            }

            override fun onNothingSelected(parent: AdapterView<*>?)
{
                // Do nothing
            }
        }
    dialog.show()
}
}
```

2. Main Activity:

```
package com.example.carproject

import android.R
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.view.animation.AnimationUtils
import com.example.carproject.databinding.ActivityMainBinding
import com.google.firebase.auth.FirebaseAuth

class MainActivity : AppCompatActivity() {

    private lateinit var mainBinding: ActivityMainBinding
    private lateinit var mAuth: FirebaseAuth



    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        mainBinding = ActivityMainBinding.inflate(layoutInflater)
        val view = mainBinding.root
        setContentView(view)
        mAuth = FirebaseAuth.getInstance()

        // Set cross-fade transition for the whole Activity
        val crossFadeAnimation = AnimationUtils.loadAnimation(this,
R.anim.fade_in)
        crossFadeAnimation.duration = 2000 // 2 seconds duration
findViewById<View>(R.id.content).startAnimation(crossFadeAnima
tion)

        mainBinding.btngetstarted.setOnClickListener {
            val intent = Intent(this@MainActivity, SignUp::class.java)
            startActivity(intent)
        }
        mainBinding.btnsignin.setOnClickListener {
            val intent = Intent(this@MainActivity, SignIn::class.java)
            startActivity(intent)
        }
        // keep user signIn
        if (mAuth.currentUser != null &&
mAuth.currentUser!!.isEmailVerified) {
            //val user = mAuth.currentUser
            val intent = Intent(this@MainActivity, Dashboard::class.java)
            startActivity(intent)
            finish()
        } else {
            // DO NOTHING
            //Toast.makeText(this, "Please verify your email.",
Toast.LENGTH_SHORT).show()
        }
    }
}
```

3. Map Maker:

```
package com.example.carproject

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

import  com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import  com.google.android.gms.maps.OnMapReadyCallback
import   com.google.android.gms.maps.SupportMapFragment
import com.google.android.gms.maps.model.LatLng
import com.google.android.gms.maps.model.MarkerOptions
import
com.example.carproject.databinding.ActivityMapsmarkerBinding

class MapsMarker : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var mMap: GoogleMap
    private lateinit var binding: ActivityMapsmarkerBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityMapsmarkerBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // Obtain the SupportMapFragment and get notified when the map
is ready to be used.
        val mapFragment = supportFragmentManager
            .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
    }

    override fun onMapReady(p0: GoogleMap) {
        TODO("Not yet implemented")
    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
*      This is where we can add markers or lines, add listeners or move
the camera. In this case,
     * we just add a marker near Sydney, Australia.
*      If Google Play services is not installed on the device, the user will
be prompted to install
*      it inside the SupportMapFragment. This method will only be
triggered once the user has
     * installed Google Play services and returned to the app.
     */
//    override fun onMapReady(googleMap: GoogleMap) {
//        mMap = googleMap
//
//        // Add a marker in Sydney and move the camera
//        val ahmedabad = LatLng(23.022505, 72.571365)
//
mMap.addMarker(MarkerOptions().position(ahmedabad).title("Mark
er in Gujarat"))
//
mMap.moveCamera(CameraUpdateFactory.newLatLng(ahmedabad))
//    }
}
```

4. My Cars:

```
package com.example.carproject

import android.content.Intent
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.ItemTouchHelper
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.example.carproject.Adapter.CarAdapter
import com.example.carproject.Models.Car
import com.example.carproject.databinding.FragmentMycarsBinding
import com.google.android.material.snackbar.Snackbar
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener

// HELP ME REMOVE ITEM OF CAR LIST USING ITEMTOUCHHEKLPER BELOW CODE SNIPPET
class MyCars : Fragment() {

    private lateinit var mycarsBinding: FragmentMycarsBinding
    private lateinit var carAdapter: CarAdapter
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        mycarsBinding      =      FragmentMycarsBinding.inflate(inflater,
container, false)

        // add cars intent
        mycarsBinding.btnAddCars.setOnClickListener {
            val intent = Intent(requireContext(), AddCar::class.java)
            startActivity(intent)
        }

        //display  car cards
        displayCars()
        return mycarsBinding.root
    }

    private fun displayCars() {
        val user = FirebaseAuth.getInstance().currentUser
        val databaseReference =

FirebaseDatabase.getInstance().getReference("Customer").child(user!
!.uid).child("Cars")
        val carList = mutableListOf<Car>()
        databaseReference.addValueEventListener(object                    :
ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                for (carSnapshot in snapshot.children) {
```

```
            val carBrand = carSnapshot.child("Car_Brand").value as?
String ?: ""
            val carType = carSnapshot.child("Car_Type").value as?
String ?: ""
            val fuelType = carSnapshot.child("Car_Fuel").value as?
String ?: ""
            val carName = carSnapshot.child("Car_Name").value as?
String ?: ""
            val car = Car(carName, carType, fuelType, carBrand)
            carList.add(car)
        }
        carAdapter = CarAdapter(carList)

        ItemTouchHelper(object                                    :
ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT) {
            override fun onMove(
                recyclerView: RecyclerView,
                viewHolder: RecyclerView.ViewHolder,
                target: RecyclerView.ViewHolder
            ): Boolean {
                return false
            }

            override           fun           onSwiped(viewHolder:
RecyclerView.ViewHolder, direction: Int) {
                val         deletedCar:         Car            =
carList[viewHolder.adapterPosition]
                val position = viewHolder.adapterPosition
                carList.removeAt(position)
                carAdapter.notifyItemRemoved(position)
                Snackbar.make(
                    mycarsBinding.rvLayout,
                    "Deleted ${deletedCar.Car_Type}",
                        Snackbar.LENGTH_LONG
                    ).setAction("Undo") {
                        carList.add(position, deletedCar)
                        carAdapter.notifyItemInserted(position)
                    }.show()
                }


            }).attachToRecyclerView(mycarsBinding.rvLayout)

            mycarsBinding.rvLayout.layoutManager                    =
LinearLayoutManager(requireContext())
            mycarsBinding.rvLayout.adapter = carAdapter
        }

        override fun onCancelled(error: DatabaseError) {
            // Handle error
        }
    })
}

companion object {
    @JvmStatic
    fun newInstance() = MyCars()
}
}


// Attach a listener to get the car details for the current user
//              databaseReference.addValueEventListener(object    :
ValueEventListener {
//        override fun onDataChange(snapshot: DataSnapshot) {
```

```
// Clear the card view
//                mycarsBinding.carsContainer.removeAllViews()
//
//                 // Inflate the layout for the card view and populate it with the car details
//                for (carSnapshot in snapshot.children) {
//                    val carData = carSnapshot.getValue(Car::class.java)
//                      val carView = layoutInflater.inflate(R.layout.car_card, null)
//
//                    carView.findViewById<TextView>(R.id.car_name).text = carData?.carName
//                   carView.findViewById<TextView>(R.id.car_brand).text = carData?.carBrand
//                  carView.findViewById<TextView>(R.id.car_model).text = carData?.carModel
//               carView.findViewById<TextView>(R.id.car_number).text = carData?.carNumber
//
//                    mycarsBinding.carsContainer.addView(carView)
//
```

Chapter 5:

Testing

5.1 Test Strategy:

1. Test Objectives:
    - Ensure all features and functionalities of the On Road Vehicle Breakdown Help Assistance application meet the specified requirements.
    - Identify and mitigate defects, errors, and inconsistencies in the application.
    - Validate the usability, accessibility, and user experience of the application.
    - Verify the security, privacy, and compliance aspects of the application.
    - Assess the performance, scalability, and reliability of the application under varying conditions.

2. Test Phases:
    - Unit Testing: Individual components and modules of the application are tested in isolation to ensure they function correctly.
    - Integration Testing: Integrated components are tested to ensure they interact seamlessly and produce the expected results.
    - System Testing: The entire application is tested as a whole to validate end-to-end functionality, usability, and performance.
    - Acceptance Testing: The application is tested against user acceptance criteria to ensure it meets user expectations and business requirements.

3. Test Methodologies:
    - Manual Testing: Testers manually execute test cases to validate application functionality, usability, and user experience.
    - Automated Testing: Automated test scripts are used to perform repetitive and regression testing tasks, improving efficiency and accuracy.
    - Exploratory Testing: Testers explore the application dynamically to uncover defects, usability issues, and edge cases not covered by scripted tests.
    - Load Testing: The application is subjected to simulated loads to evaluate performance, scalability, and resource utilization under high traffic conditions.

4. Test Coverage:
    - Functional Testing: Validate all functional requirements specified for the application, including user registration, appointment booking, service management, and payment processing.
    - Non-Functional Testing: Assess non-functional aspects such as usability, security, performance, compatibility, and regulatory compliance.
    - Edge Case Testing: Test application behavior in edge cases, including invalid inputs, boundary conditions, and error handling scenarios.
    - User Experience Testing: Evaluate the application's usability, accessibility, and user interface design to ensure a positive user experience.
    - Security Testing: Conduct security testing to identify vulnerabilities, threats, and risks to the application's data and infrastructure.

5. Test Environment:
    - Utilize testing environments that closely resemble the production environment to ensure accurate testing results.
    - Use virtualization and containerization technologies to create isolated test environments for testing different components and configurations.

6. Test Tools:
    - Test Management Tools: Use tools like Jira, TestRail, or HP ALM for test case management, defect tracking, and reporting.
    - Automation Tools: Employ automation tools such as Selenium WebDriver, Appium, or Cypress for automating test scripts and regression testing.
    - Performance Testing Tools: Utilize tools like Apache JMeter, LoadRunner, or Gatling for load testing and performance monitoring.

7. Reporting and Documentation:
    - Document test plans, test cases, test results, and defects encountered during testing.
    - Generate test reports summarizing test coverage, test execution status, and any issues identified during testing.
    - Provide stakeholders with regular updates on testing progress, including milestones achieved and risks identified.

8. Test Team Collaboration:
    - Foster collaboration and communication among cross-functional teams, including developers, testers, designers, and product owners.
    - Conduct regular meetings, reviews, and feedback sessions to ensure alignment on testing objectives, priorities, and outcomes.

9. Continuous Improvement:
    - Continuously monitor and evaluate the effectiveness of testing processes, methodologies, and tools.
    - Incorporate feedback from testing activities to improve test coverage, efficiency, and effectiveness.
    - Identify lessons learned and best practices to inform future testing efforts and enhance overall quality assurance practices.

5.2 Unit Test Plan:

1. Objective:
   - The objective of unit testing is to validate the functionality, correctness, and reliability of individual units or components of the On Road Vehicle Breakdown Help Assistance application.

2. Scope:
   - The unit test plan covers the testing of all functional components, modules, and classes of the On Road Vehicle Breakdown Help Assistance application, including backend APIs, frontend components, and database operations.

3. Testing Approach:
   - White-box Testing: Unit tests will be designed based on the internal structure, code logic, and business rules of each component.
   - Test-Driven Development (TDD): Follow the TDD approach by writing tests before implementing the code to ensure comprehensive test coverage.

4. Test Environment:
   - Utilize development or testing environments that closely resemble the production environment, including necessary dependencies and configurations.

5. Testing Tools:
   - Use testing frameworks and libraries compatible with the technology stack of the On Road Vehicle Breakdown Help Assistance application, such as JUnit, Mockito, Jest, or PyTest.

6. Test Cases:
    - Develop unit test cases for each functional unit or method, covering both positive and negative scenarios, edge cases, and boundary conditions.
    - Test cases should verify input validation, error handling, exception handling, and expected outcomes.

7. Test Coverage:
    - Aim for high test coverage to ensure that all critical paths and code branches are tested.
    - Target a minimum coverage threshold (e.g., 80%) to maintain code quality and reliability.

8. Test Execution:
    - Execute unit tests automatically as part of the continuous integration (CI) pipeline to ensure early detection of defects.
    - Run unit tests locally during development and before merging code changes to the main branch.

9. Test Reporting:
    - Document test results, including pass/fail status, test coverage metrics, and any defects encountered during testing.
    - Generate test reports and share them with the development team for review and analysis.

10. Test Maintenance:
    - Update unit tests as needed to accommodate changes in the application code, requirements, or design.
    - Refactor and optimize existing unit tests for improved readability, maintainability, and performance.

11. Testing Considerations:

- Concurrency Testing: Test components that handle concurrent requests or operations to ensure thread safety and avoid race conditions.
- Integration Points: Mock or stub external dependencies (e.g., databases, APIs) to isolate units and focus on testing individual functionality.
- Performance Testing: Identify and address performance bottlenecks or inefficiencies at the unit level to optimize application performance.

12. Review and Approval:

- Review the unit test plan with relevant stakeholders, including developers, testers, and project managers, to ensure alignment with project goals and objectives.
- Obtain approval from the project lead or QA manager before proceeding with test execution.

5.3 Acceptance Test Plan:

1. Objective:
   - The objective of acceptance testing is to validate that the On Road Vehicle Breakdown Help Assistance Car Service Provider Application meets user requirements, business objectives, and quality standards before deployment.

2. Scope:
   - The acceptance test plan covers the testing of end-to-end functionality, usability, and performance of the On Road Vehicle Breakdown Help Assistance application from a user's perspective.

3. Testing Approach:
   - Black-box Testing: Acceptance tests will be designed based on the application's external behavior and user interactions, without knowledge of its internal implementation.
   - User Scenario Testing: Test real-world user scenarios and workflows to ensure that the application meets user needs and expectations.

4. Test Environment:
   - Utilize staging or pre-production environments that closely resemble the production environment, including necessary configurations and datasets.

5. Testing Tools:
   - Use testing tools and frameworks suitable for conducting manual acceptance testing, such as TestRail, JIRA, or Microsoft Excel.

6. Test Cases:
   - Develop acceptance test cases based on user stories, use cases, and functional requirements specified for the On Road Vehicle Breakdown Help Assistance application.
   - Test cases should cover all critical features, workflows, and user interactions of the application.

7. Test Scenarios:
   - User Registration and Login:
     - Verify that users can register for an account and log in successfully using valid credentials.
     - Validate the functionality of password recovery and account activation processes.

   - Appointment Booking:
     - Test the process of scheduling service appointments, including selecting services, mechanics, dates, and times.
     - Verify that users receive confirmation notifications and appointment reminders.

   - Service Management:
     - Ensure administrators can manage mechanics, services, and user accounts effectively through the admin dashboard.
     - Validate the functionality of adding, editing, and deleting mechanics and services.

   - Payment Processing:
     - Test the payment process for booking service appointments, including selecting payment methods, entering payment details, and completing transactions.

        - Verify that users receive payment confirmation and receipts for successful transactions.

    - Emergency Assistance:
        - Validate the functionality of emergency location mapping and communication tools for users in critical situations.
        - Test the process of sharing location details with emergency responders and service providers.

8. Test Data:
    - Prepare test data sets that simulate real-world scenarios, including sample user accounts, service appointments, and administrative configurations.

9. Test Execution:
    - Execute acceptance tests manually by following predefined test cases and scenarios.
    - Record test results, including pass/fail status, observations, and any defects encountered during testing.

10. Defect Reporting:
    - Document and report any defects or issues encountered during acceptance testing using a standardized defect tracking system.
    - Include detailed descriptions, screenshots, and steps to reproduce each reported defect.

11. Regression Testing:

- Perform regression testing to ensure that fixes for reported defects do not introduce new issues or regressions.
- Re-run previously executed acceptance tests to validate the stability and reliability of the application.

12. Test Sign-off:

- Obtain sign-off from stakeholders, including product owners, project managers, and QA leads, to confirm the completion of acceptance testing and readiness for deployment

5.4 Test Case / Test Script:

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC001	User Registration	User is not registered	1. Navigate to the registration page	User successfully registers an account	Pass
			2. Enter valid user details (name, email, password)		
			3. Click on the "Register" button		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC002	User Login	User is registered and not logged in	1. Navigate to the login page	User successfully logs in with valid credentials	Pass
			2 Enter registered email and password		
			3. Click on the "Login" button		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC003	Add Mechanic	Administrator is logged in	1 Navigate to the "Manage Mechanics" section	Mechanic successfully added to the system	Pass
			2 Click on the "Add Mechanic" button		
			3. Enter mechanic details (name, contact, expertise)		
			4. Click on the "Save" button		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC004	Manage User	Administrator is logged in	1. Navigate to the "Manage Users" section	User details successfully updated in the system	Pass
			2. Search for the user to be managed		
			3. Edit user details (name, email, contact, etc.)		
			4. Click on the "Save" button		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC005	Register Car	User is logged in and has not registered a car	1. Navigate to the "Register Car" section	Car successfully registered to the user	Pass
			2. Enter car details (make, model, year, registration)		
			3. Click on the "Register Car" button		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC006	Manage Service	Administrator is logged in	1. Navigate to the "Manage Services" section	Service details successfully updated in the system	Pass
			2. Search for the service to be managed		
			3. Edit service details (name, description, price, etc.)		
			4. Click on the "Save" button		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC007	Book Service	User is logged in and has a registered car	1. Navigate to the "Book Service" section	Service appointment successfully booked	Pass
			2 Select desired service and mechanic		
			3. Choose preferred date and time		
			4. Click on the "Book Now" button		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC008	Get Price Detail	User is logged in	1. Navigate to the service details page	Price details for the selected service displayed	Pass
			2. Select the desired service		
			3. Verify the displayed price		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC009	Confirm Booking	User has booked a service appointment	1. Navigate to the booked services section	Service appointment confirmed successfully	Pass
			2. Select the booked appointment		
			3. Click on the "Confirm" button		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC010	Get Service Details	User is logged in and has booked a service	1 Navigate to the service details page	Details of the booked service displayed	Pass
		appointment			
			2 Select the booked service appointment		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC011	Generate Receipt	Service appointment is completed	1. Navigate to the completed appointments section	Receipt for the completed service appointment	Pass
			2. Select the completed appointment	displayed	
			3. Click on the "Generate Receipt" button		

Test Case ID	Test Case Description	Precondition	Steps	Expected Result	Pass/Fail
TC012	Payment	Service appointment is completed	1. Navigate to the completed appointments section	Payment for the completed service appointment	Pass
			2. Select the completed appointment	completed successfully	
			3. Click on the "Make Payment" button		

Defect report / Test Log:

Defect ID	Description	Severity	Priority	Status	Reported By	Assigned To
DEF001	Unable to register user account	High	High	Open	Tester	Developer A
DEF002	Error message not displayed on login failure	Medium	Medium	Open	Tester	Developer B
DEF003	Appointment booking date selection issue	High	High	Closed	Tester	Developer C
DEF004	Payment processing error	High	High	Open	Tester	Developer A
DEF005	Emergency location mapping not working	High	High	Open	Tester	Developer B

Chapter 6:

Limitation of Proposed System

1. Dependency on Internet Connectivity:
   - The On Road Vehicle Breakdown Help Assistance application heavily relies on internet connectivity for users to access its features.
   - In areas with poor or no internet connectivity, users may experience difficulties in using the application, which could impact their ability to book appointments or access emergency assistance.

2. Limited Accessibility for Technologically Challenged Users:
   - Elderly or technologically challenged users may find it difficult to navigate through the application's interface or perform tasks such as registering cars, booking appointments, or making payments online.
   - This limitation could potentially exclude a segment of the user population from utilizing the application effectively.

3. Data Privacy and Security Concerns:
   - Storing sensitive user information, such as personal details, car registration data, and payment information, within the application poses inherent data privacy and security risks.
   - Without robust security measures in place, the system may be vulnerable to data breaches, identity theft, or unauthorized access, compromising user trust and confidence.

4. Dependency on Third-Party Services:
   - The integration of third-party services, such as Google Maps for emergency location mapping or payment gateways for processing transactions, introduces a dependency on external providers.
   - Any disruptions or changes to these services could impact the functionality and reliability of the On Road Vehicle Breakdown Help Assistance application.

5. Scalability Challenges:
    - As the user base and volume of transactions increase over time, the scalability of the On Road Vehicle Breakdown Help Assistance application may become a concern.
    - Inadequate infrastructure or architectural limitations could lead to performance bottlenecks, slower response times, and degraded user experience during peak usage periods.

6. Limited Offline Functionality:
    - The On Road Vehicle Breakdown Help Assistance application may lack offline functionality, making it inaccessible to users when they are not connected to the internet.
    - Users may encounter difficulties in accessing essential features or information, especially during emergencies or in remote locations with poor connectivity.

7. Platform Compatibility Issues:
    - Ensuring compatibility with all major web browsers and mobile devices is challenging due to the diverse range of platforms, screen sizes, and operating systems available in the market.
    - The application may encounter compatibility issues or display inconsistencies across different devices, impacting user experience and satisfaction.

8. Complexity of Maintenance and Updates:
    - Maintaining and updating the On Road Vehicle Breakdown Help Assistance application to address bugs, add new features, or comply with regulatory changes requires ongoing effort and resources.

- Managing the complexity of codebase, dependencies, and integration points may pose challenges for developers and administrators, potentially leading to delays or disruptions in service.

9. User Adoption and Behavior:
   - User adoption and behavior play a crucial role in the success of the On Road Vehicle Breakdown Help Assistance application.
   - Encouraging users to adopt the application, engage with its features regularly, and provide feedback for improvement may require targeted marketing efforts, incentives, or user education programs.

10. Competitive Landscape:
    - In a competitive market landscape, the On Road Vehicle Breakdown Help Assistance application may face stiff competition from existing car service providers, as well as new entrants offering similar or enhanced features.
    - Differentiating the application and retaining users in such a competitive environment can be challenging.

Chapter 7:

Proposed Enhancement

1. **Offline Mode Support:**
    - Implement an offline mode feature that allows users to access essential functionalities, such as viewing upcoming appointments, emergency assistance, and service history, even when they are not connected to the internet.
    - Offline data synchronization capabilities can ensure that users can continue to use the application seamlessly in areas with poor connectivity.

2. **Enhanced User Onboarding and Education:**
    - Develop interactive tutorials, guides, or tooltips within the application to onboard new users and educate them about its features and functionalities.
    - Providing clear instructions and tips can help users, especially those who are less tech-savvy, navigate the application more effectively and make the most out of its capabilities.

3. **Improved Accessibility Features:**
    - Enhance accessibility features within the application to cater to users with disabilities or impairments.
    - This includes support for screen readers, keyboard navigation, color contrast adjustments, and text resizing options.
    - Ensuring compliance with accessibility standards such as WCAG (Web Content Accessibility Guidelines) can make the application more inclusive and user-friendly.

4. **Integration with Wearable Devices:**
    - Integrate the On Road Vehicle Breakdown Help Assistance application with wearable devices such as smartwatches or fitness trackers to provide users with real-time notifications, alerts, and updates about their car's maintenance status, upcoming appointments, and emergency assistance.
    - This seamless integration can enhance user convenience and promote proactive vehicle maintenance.

5. **Advanced Security Measures:**
    - Strengthen security measures within the application to protect user data and ensure privacy.
    - Implement features such as two-factor authentication, biometric authentication, encryption of sensitive information, and regular security audits to mitigate risks of data breaches and unauthorized access.

6. **Personalized Recommendations and Alerts:**
    - Utilize machine learning algorithms and data analytics to analyze user behavior, preferences, and vehicle usage patterns.
    - Based on this analysis, provide personalized recommendations for maintenance schedules, service packages, and relevant offers.
    - Additionally, send proactive alerts and reminders to users about upcoming service appointments, vehicle recalls, or critical maintenance tasks.

7. **Integration with Smart Vehicle Technologies:**
    - Explore integration with emerging smart vehicle technologies, such as connected car platforms and onboard diagnostics systems.
    - By leveraging data from these technologies, the On Road Vehicle Breakdown Help Assistance application can offer enhanced vehicle monitoring capabilities, predictive maintenance insights, and remote diagnostic features, enabling users to better manage their vehicles' health and performance.

8. **Gamification and Rewards Program:**
    - Introduce gamification elements and a rewards program within the application to incentivize user engagement and loyalty.
    - Users can earn points, badges, or discounts by completing certain tasks, booking regular service appointments, referring friends, or participating in community challenges.
    - This gamified approach can foster a sense of achievement and encourage users to interact more frequently with the application.

9. **Community and Social Features:**
    - Create a community platform within the application where users can connect with other car owners, share tips, experiences, and recommendations related to vehicle maintenance and repairs.
    - Encourage user-generated content, discussions, and peer-to-peer support to build a vibrant and engaged user community around the On Road Vehicle Breakdown Help Assistance application.

10. **Continuous Performance Optimization:**
    - Continuously monitor and optimize the performance of the application to ensure fast response times, smooth navigation, and seamless user experience across different devices and platforms.
    - Conduct regular performance testing, identify areas for improvement, and implement optimizations to enhance the application's speed, reliability, and scalability.

Chapter 8: Conclusion

1. Technical Conclusion:
    - The technical feasibility of the proposed system is high, considering the availability of skilled developers and the compatibility of chosen technologies with the project requirements.
    - Integration with third-party services like Google Maps API and Firebase Realtime Database is feasible, enabling the implementation of essential features such as location sharing and real-time data synchronization.

2. Operational Conclusion:
    - The operational impact of the new system is manageable, with adequate training and support for users and administrators.
    - The system aligns with existing operational workflows, enhancing efficiency and streamlining car maintenance and repair processes.

3. Economic Conclusion:
    - The economic feasibility of the project is favorable, with a positive return on investment expected based on the projected benefits and cost analysis.
    - Development costs, server expenses, and maintenance fees are justifiable considering the potential market demand and revenue generation opportunities.

4. Legal and Regulatory Conclusion:
    - The proposed system complies with relevant laws and regulations, including data privacy and security requirements.
    - Intellectual property rights are respected, ensuring that the development and use of the system do not infringe on any existing patents or copyrights.

5. Schedule Conclusion:
    - The development timeline is realistic, allowing sufficient time for feature implementation, testing, and deployment.
    - Market timing aligns with industry trends, maximizing the potential for market penetration and user adoption.

6. Resource Conclusion:
    - Human resources, hardware, and software resources are available or can be obtained within the project timeline, ensuring adequate support for development and maintenance activities.

7. Market Conclusion:
    - Market research indicates a demand for car service provider applications, with potential competitors offering similar solutions.
    - User acceptance is likely, as the application addresses common pain points for car owners and provides valuable features for managing vehicle maintenance and repairs.

8. Security Conclusion:
    - The security measures implemented in the system are robust, protecting user data and ensuring secure authentication mechanisms.
    - Data privacy concerns are addressed, enhancing user trust and confidence in the application.

9. Environmental and Social Conclusion:
    - The environmental impact of the system is minimal, with considerations for energy efficiency and resource conservation.
    - Social implications are positive, as the application improves accessibility to car maintenance services and promotes safer driving practices through features like emergency assistance.

Chapter 9:

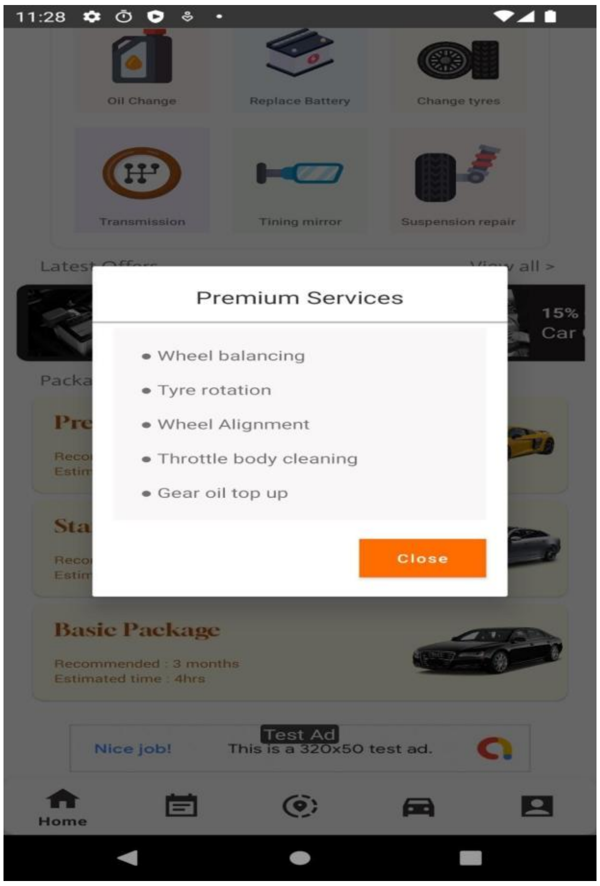
Bibliography

9. Bibliography:

https://developer.android.com/

https://firebase.google.com/

Chapter 10:
User Manual

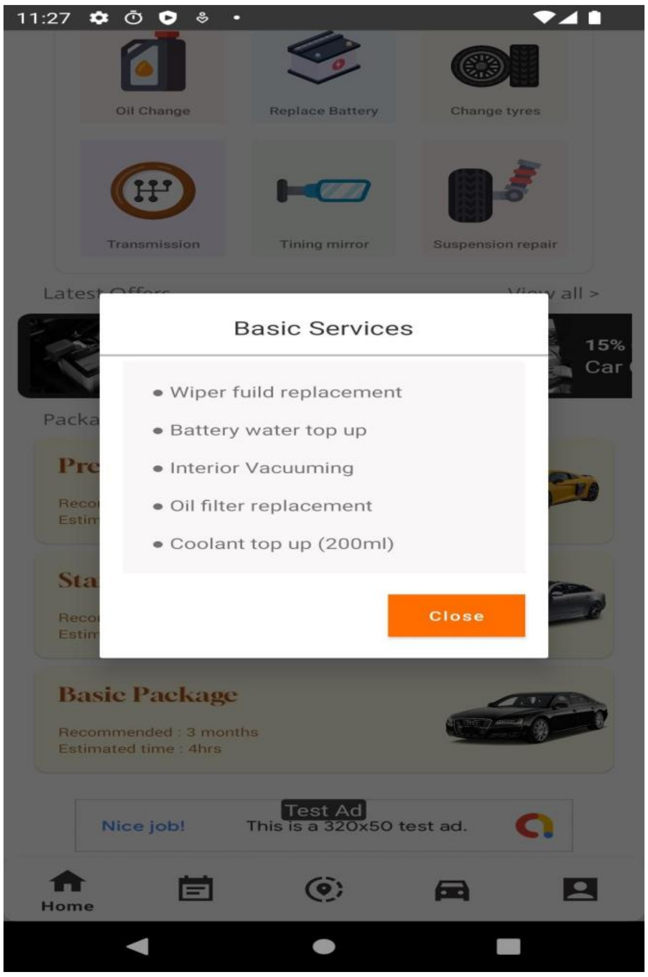


Premium Services

- Wheel balancing
- Tyre rotation
- Wheel Alignment
- Throttle body cleaning
- Gear oil top up

Close

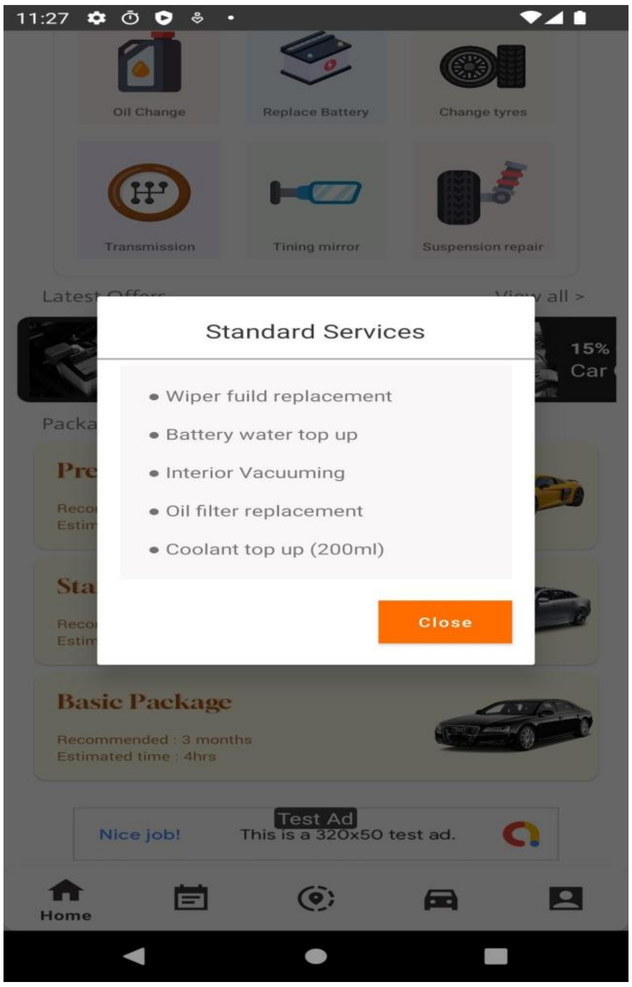
- **Description:**The premium service page on On Road Vehicle Breakdown Help Assistance offers an elevated experience for car owners seeking exceptional care for their vehicles. With a focus on quality and convenience, our premium services provide comprehensive maintenance and repair solutions tailored to meet the unique needs of each customer. From advanced detailing and diagnostics to specialized treatments and exclusive perks, our premium service offerings ensure that your car receives the highest level of care and attention it deserves. With expert technicians, state-of-the-art facilities, and personalized service packages, On Road Vehicle Breakdown Help Assistance's premium service page is your destination for unparalleled automotive care and satisfaction.



Basic Services

- Wiper fuild replacement
- Battery water top up
- Interior Vacuuming
- Oil filter replacement
- Coolant top up (200ml)

- **Description:** The service page in the On Road Vehicle Breakdown Help Assistance app offers a streamlined experience for users to explore and select from a range of essential car maintenance and repair services. With clear and concise listings, users can easily find services such as oil changes, tire rotations, brake inspections, and more. Each service listing provides key details such as pricing, service duration, and a brief description of the service offered. Users can quickly book appointments for their desired services, ensuring their vehicles receive the necessary care to stay safe and reliable on the road.

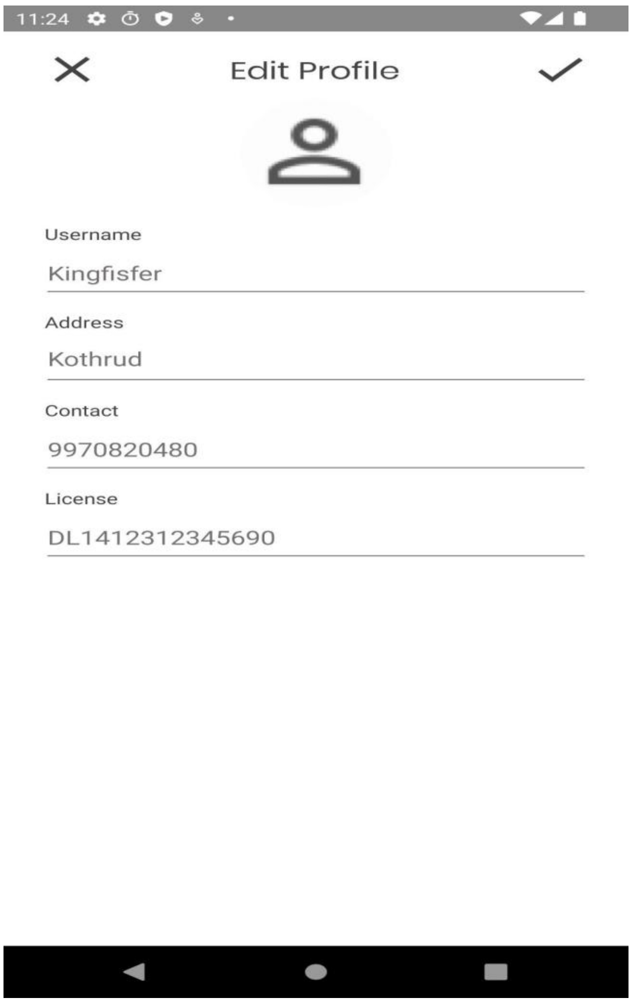


Standard Services

- Wiper fuild replacement
- Battery water top up
- Interior Vacuuming
- Oil filter replacement
- Coolant top up (200ml)

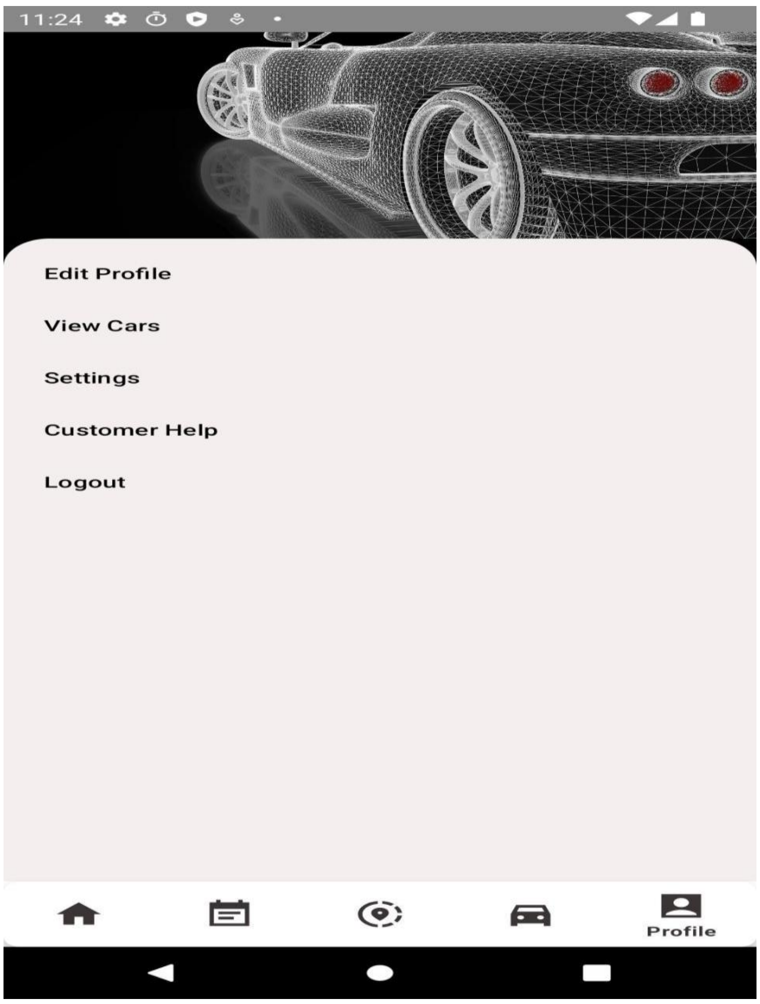
Close

➢ **Description:** The standard service page on On Road Vehicle Breakdown Help Assistance presents users with a comprehensive array of automotive maintenance and repair options tailored to meet diverse vehicle needs. From routine tasks like oil changes and tire rotations to more intricate services such as engine diagnostics and electrical system checks, users can browse through detailed descriptions and transparent pricing structures. With easy appointment booking functionality directly from the page, users can efficiently schedule services, ensuring their vehicles remain in top condition for safe and reliable driving experiences**.**

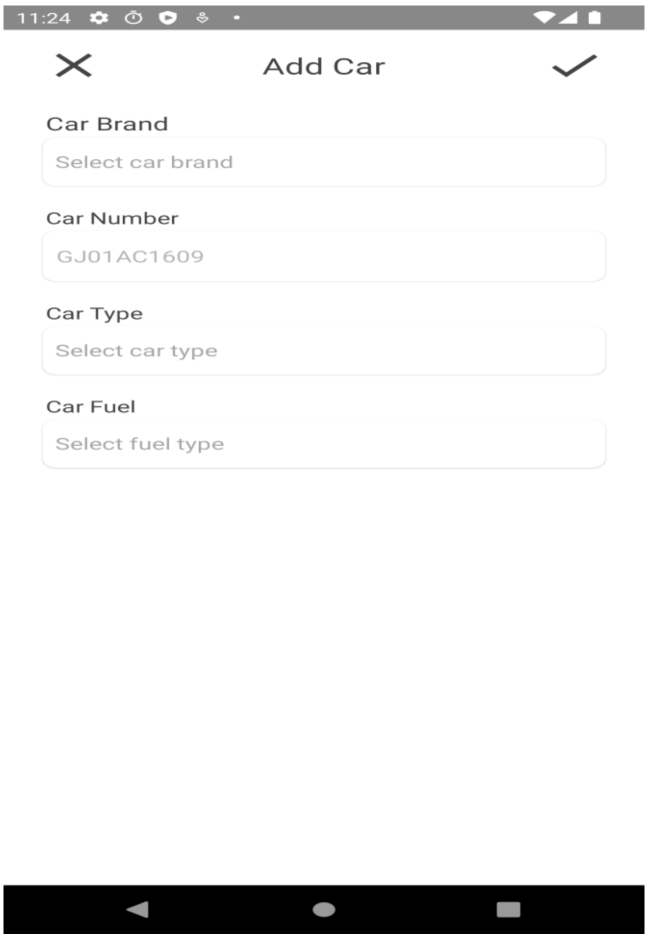


Username
Kingfisfer
Address
Kothrud
Contact
9970820480
License
DL1412312345690

- **Description**: The edit profile page on On Road Vehicle Breakdown Help Assistance empowers users to manage and customize their personal information with ease. Through intuitive user interfaces, users can update their contact details, address, and preferences effortlessly. Additionally, the page offers options to modify profile pictures and other relevant information, ensuring accuracy and relevance. By providing a seamless experience for users to adjust their profile settings, On Road Vehicle Breakdown Help Assistance ensures that user accounts remain up-to-date and tailored to individual preferences.

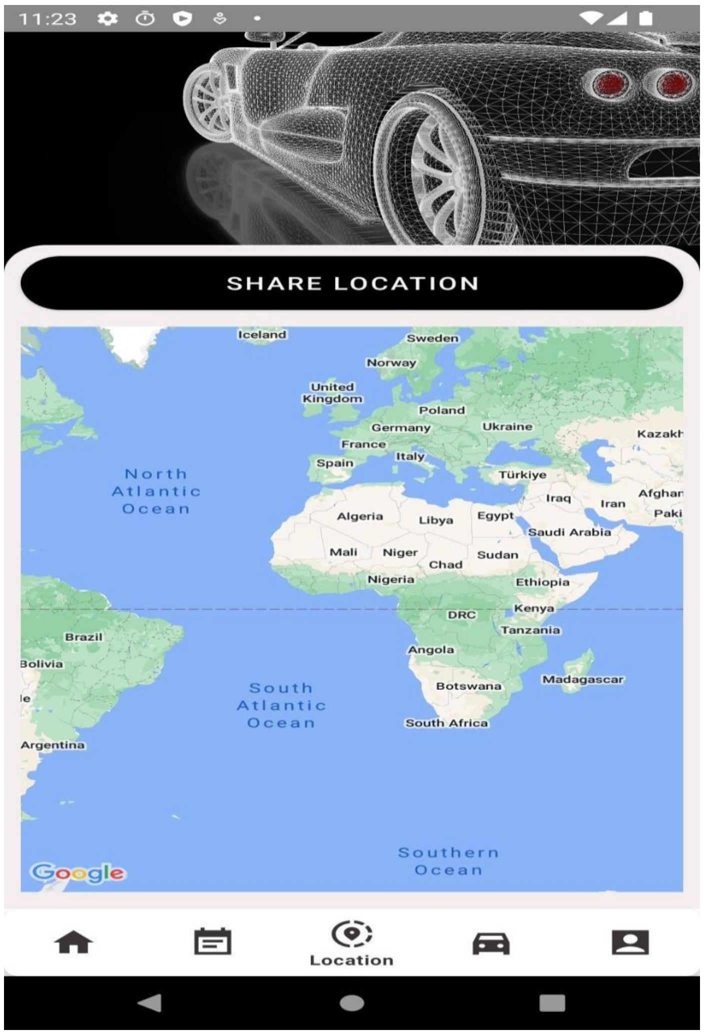


- **Description:** The user profile page on On Road Vehicle Breakdown Help Assistance serves as a centralized hub for users to access and manage their account information conveniently. Here, users can view and edit their personal details, including contact information, address, and vehicle preferences. Additionally, the user profile page offers insights into past service history and upcoming appointments, allowing users to stay informed about their vehicle maintenance schedules. With intuitive navigation and comprehensive features, On Road Vehicle Breakdown Help Assistance's user profile page ensures a personalized and seamless experience for users to maintain their account settings and track their automotive needs.

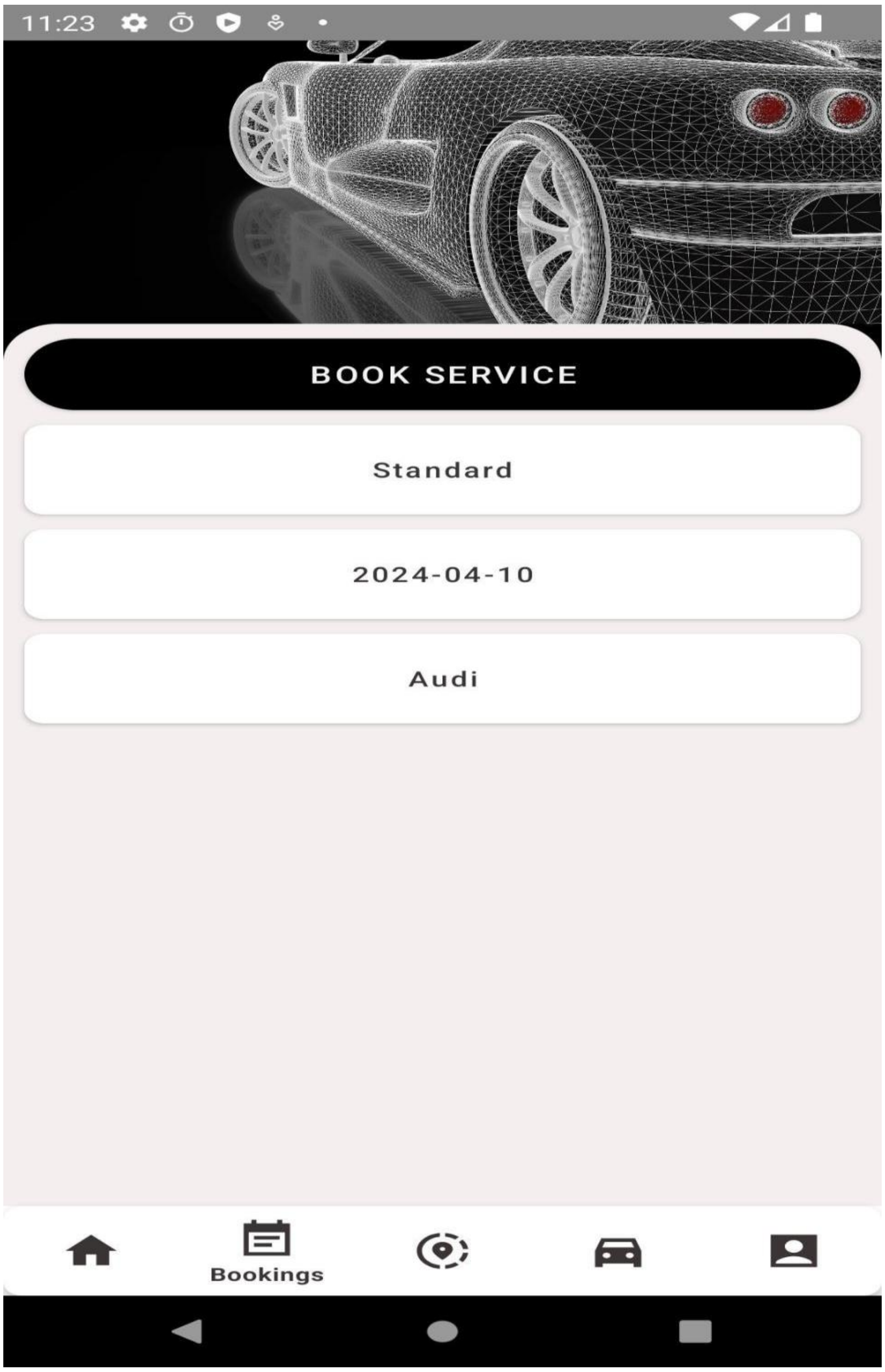


Car Brand
Select car brand
Car Number
GJ01AC1609
Car Type
Select car type
Car Fuel
Select fuel type

- **Description**: The "Add Car" page on On Road Vehicle Breakdown Help Assistance streamlines the process of incorporating new vehicles into users' profiles. With user-friendly interfaces, individuals can input essential details about their vehicles, including make, model, year, and license plate information. Additionally, the page offers options to upload images or provide additional notes to personalize each car entry. By facilitating efficient data entry and customization, On Road Vehicle Breakdown Help Assistance's "Add Car" page ensures that users can effortlessly manage their fleet of vehicles within the application, optimizing their automotive experience.



- **Description:** The location page on On Road Vehicle Breakdown Help Assistance enables users to share their current location seamlessly. Through integrated maps and intuitive interfaces, users can pinpoint their whereabouts and provide precise location details to service providers or emergency responders. Additionally, the page offers functionalities to input additional context or instructions to enhance communication. With streamlined sharing capabilities, On Road Vehicle Breakdown Help Assistance's location page ensures swift and accurate assistance during critical situations, enhancing user safety and peace of mind.

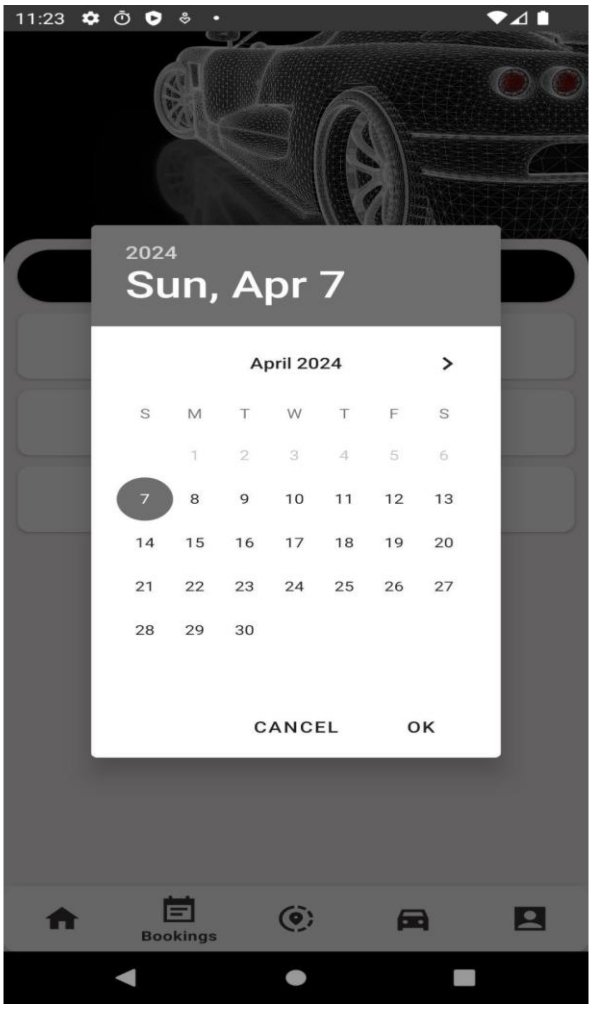


Standard

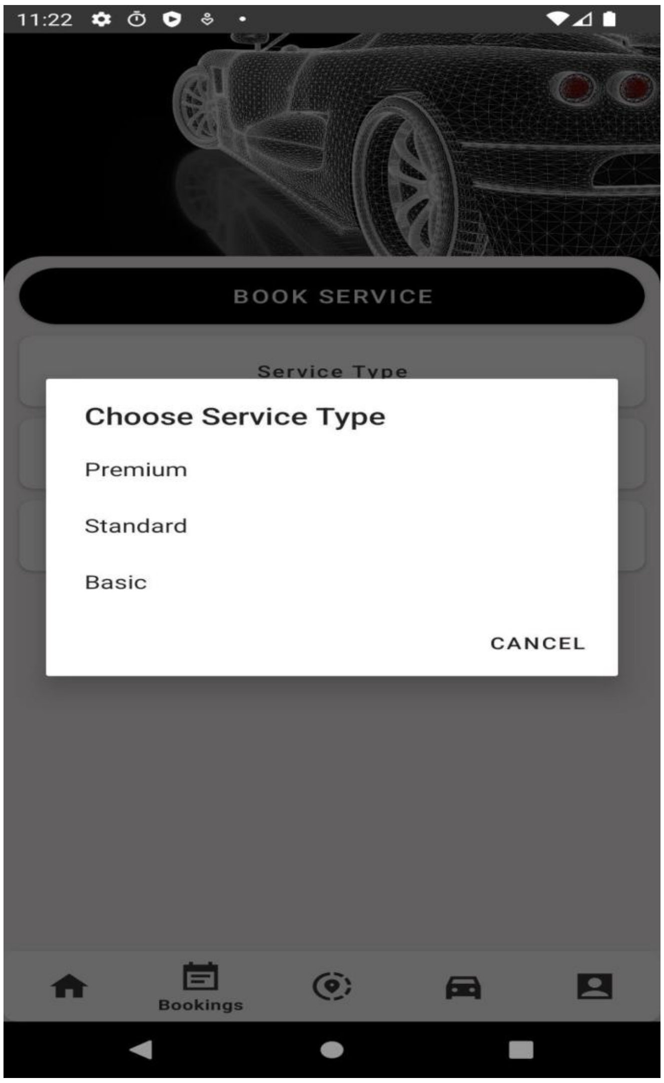
2024-04-10

Audi

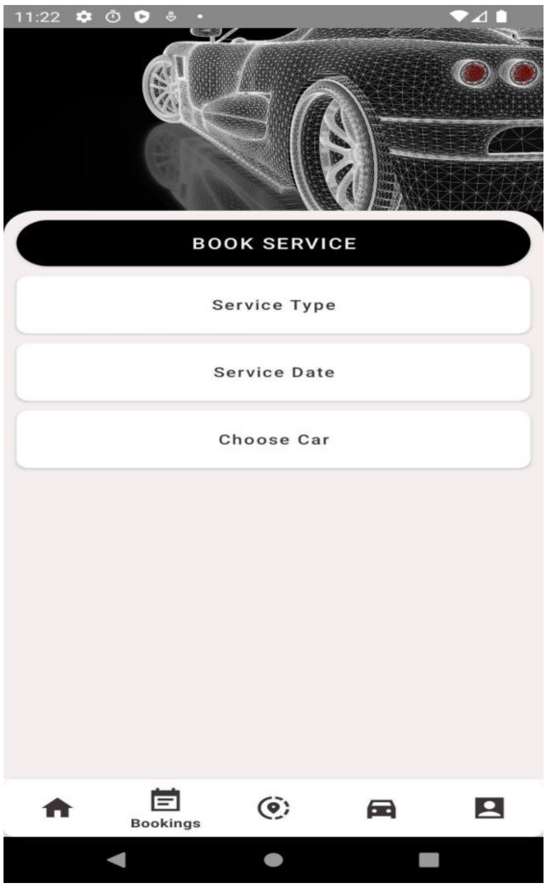
➢ **Description:** a page which shows the information of the booking done by customer



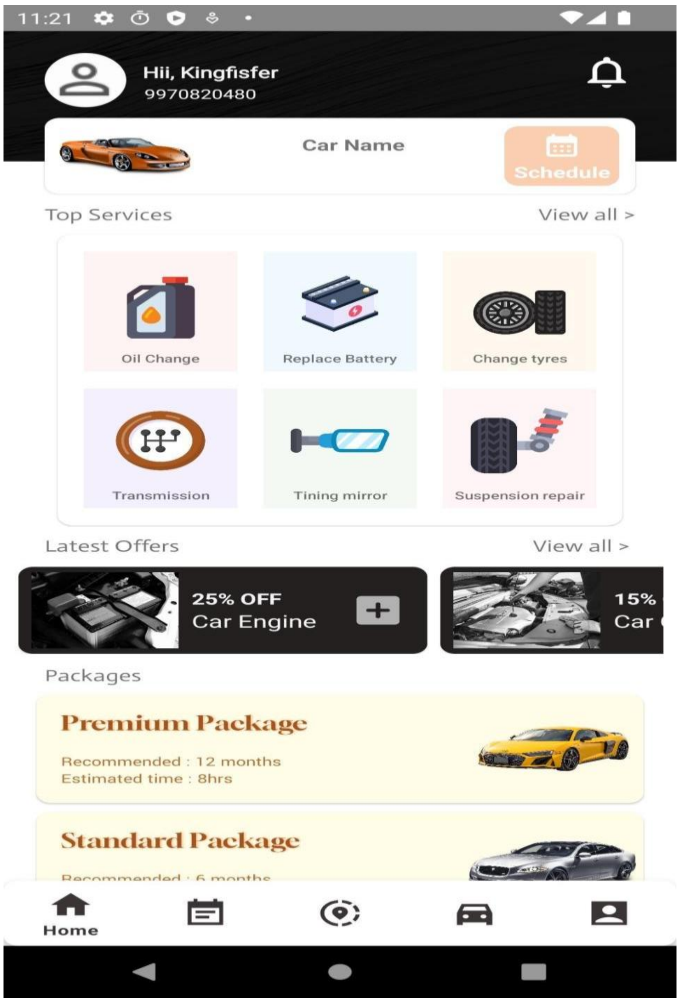
➢ **Description:** The date page on On Road Vehicle Breakdown Help Assistance facilitates the scheduling of appointments by allowing users to select their preferred date and time for service. Through a user-friendly interface, individuals can easily navigate calendar options and choose available time slots that best fit their schedules. With clear visibility of available dates and times, On Road Vehicle Breakdown Help Assistance's date page ensures a seamless booking experience, enabling users to efficiently plan their vehicle maintenance appointments with convenience.

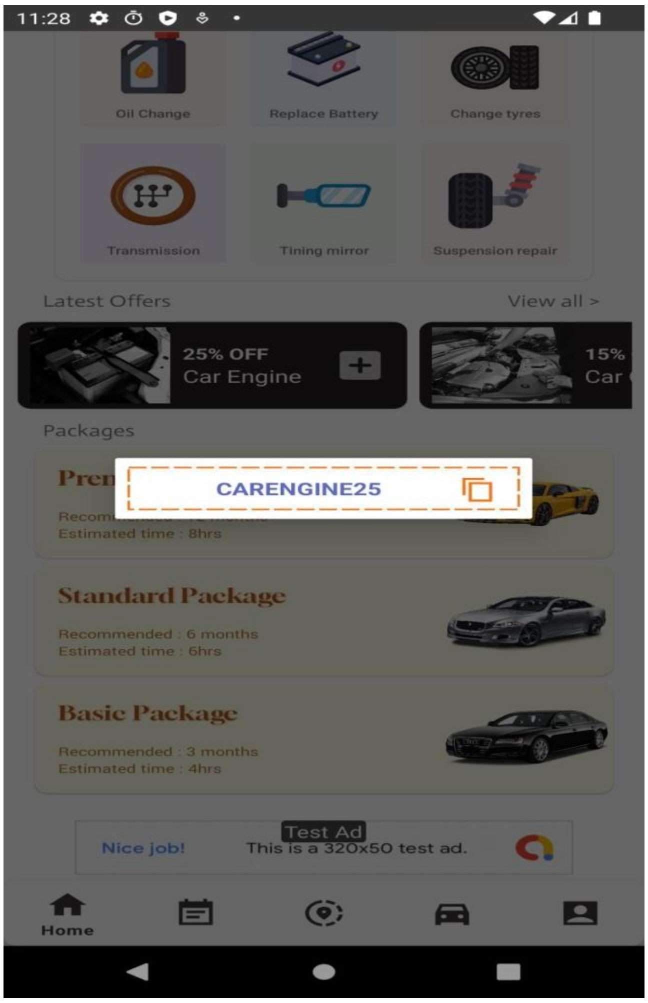


- **Description:** The booking service page on On Road Vehicle Breakdown Help Assistance simplifies the process of scheduling vehicle maintenance and repairs. Through an intuitive interface, users can select from a variety of services and specify their preferred date and time for appointments. With transparent pricing and service descriptions available, users can make informed decisions about their vehicle care needs. By offering streamlined booking functionalities, On Road Vehicle Breakdown Help Assistance's booking service page ensures convenience and efficiency for users, helping them keep their vehicles in optimal condition with ease.

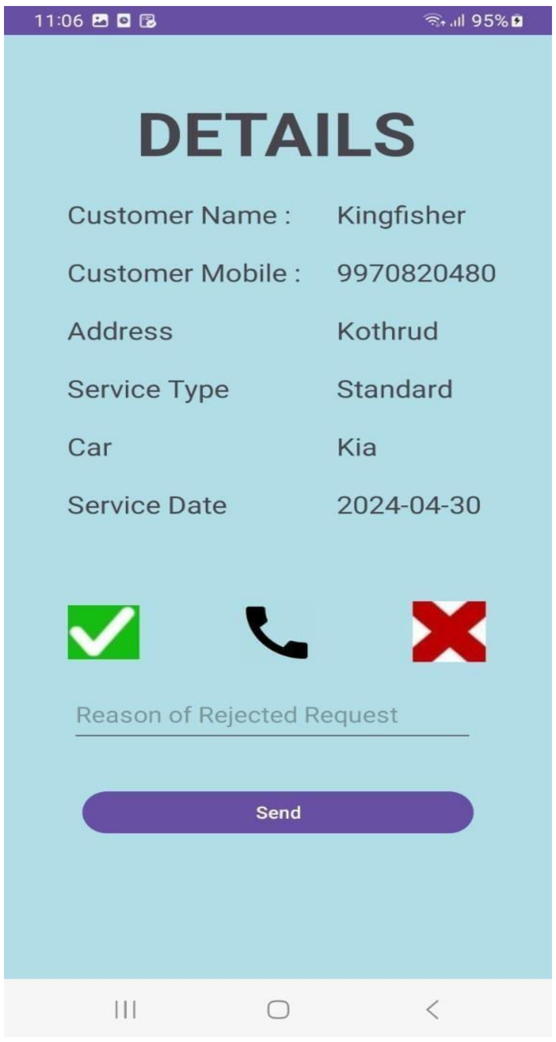


- **Description:** The customer booking service page on On Road Vehicle Breakdown Help Assistance offers users a straightforward and efficient way to schedule their vehicle maintenance appointments. With a user- friendly interface, individuals can easily browse through available services, select their desired options, and choose a convenient date and time for their appointment. Transparent pricing and detailed service descriptions empower users to make informed decisions about their vehicle care needs. By providing a seamless booking process, On Road Vehicle Breakdown Help Assistance's customer booking service page ensures convenience and satisfaction for users, helping them keep their vehicles well-maintained with ease
- navigation throughout the application.





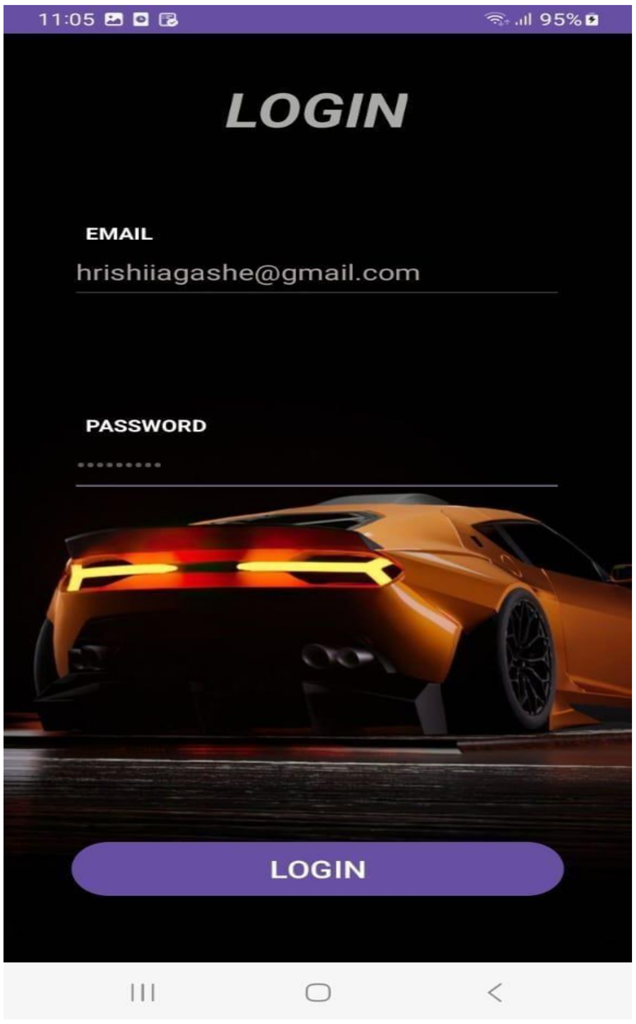
➢ **Description:** The home page of On Road Vehicle Breakdown Help Assistance serves as a centralized hub where users can access essential features and information related to their vehicle maintenance needs. With intuitive navigation, users can easily explore a variety of services, view the latest offers, and browse through available service packages. Through clear and concise displays, On Road Vehicle Breakdown Help Assistance's home page provides users with valuable insights and options to address their automotive requirements effectively. By offering a comprehensive overview of available services and promotions, the home page ensures a user-friendly experience that promotes informed decision- making and seamless navigation throughout the application.



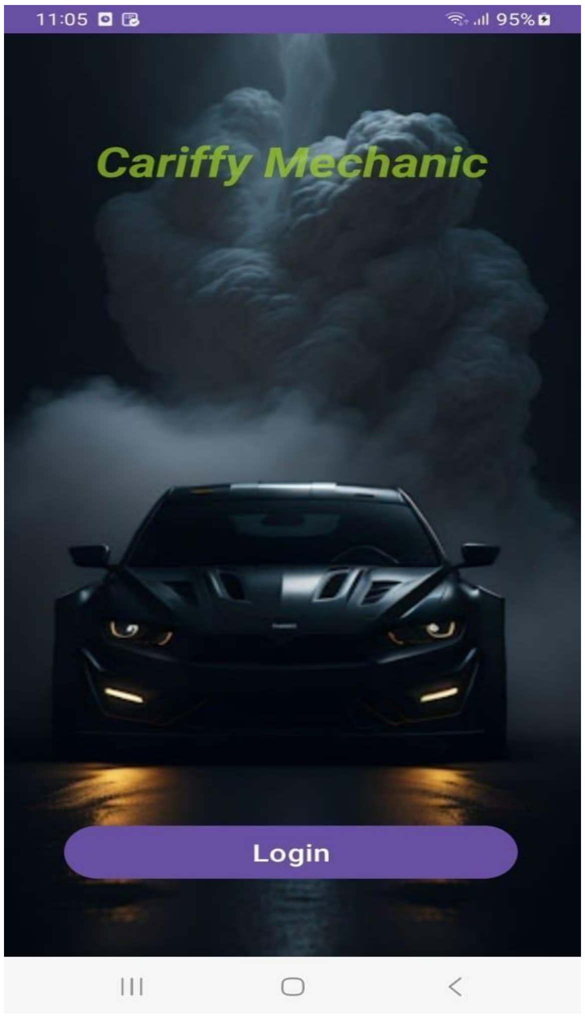
DETAILS

Customer Name :	Kingfisher
Customer Mobile :	9970820480
Address	Kothrud
Service Type	Standard
Car	Kia
Service Date	2024-04-30

➢ **Description:** The user request page on On Road Vehicle Breakdown Help Assistance enables users to submit and manage their service requests conveniently. Through an intuitive interface, users can input details about the type of service needed, preferred appointment dates, and any additional notes or requirements. Once submitted, requests are processed by service providers, and users can track the status of their requests in real-time. With transparent communication and efficient management features, On Road Vehicle Breakdown Help Assistance's user request page ensures a streamlined process for users to receive timely and reliable automotive services.



- **Description:** The mechanic login page on On Road Vehicle Breakdown Help Assistance provides mechanics with secure access to their accounts, allowing them to manage service requests and appointments efficiently. Through a user-friendly interface, mechanics can enter their credentials to log in and gain access to the platform's features. Once logged in, mechanics can view assigned tasks, update service statuses, and communicate with customers as needed. With robust authentication measures in place, On Road Vehicle Breakdown Help Assistance's mechanic login page ensures that mechanics can securely access the system and fulfill their responsibilities effectively.



- **Description:** The mechanic login page on On Road Vehicle Breakdown Help Assistance provides mechanics with secure access to their accounts, allowing them to manage service requests and appointments efficiently. Through a user-friendly interface, mechanics can enter their credentials to log in and gain access to the platform's features. Once logged in, mechanics can view assigned tasks, update service statuses, and communicate with customers as needed. With robust authentication measures in place, On Road Vehicle Breakdown Help Assistance's mechanic login page ensures that mechanics can securely access the system and fulfill their responsibilities effectively.