Garagat MVP Development – Scope of Work

Project Overview

Garagat is envisioned as a mobile-first car care platform in the UAE, offering on-demand vehicle services through customer mobile apps and garage/fleet web portals. Customers can book services (detailing, maintenance, repairs, etc.) and securely pay within the app, while partner garages manage incoming orders and corporate fleet clients monitor their vehicles' service needs. The client's branding guidelines will be applied to all interfaces. This Statement of Work (SOW) defines the MVP scope and a ~3-month development plan. It includes essential features like vehicle pickup and delivery scheduling (without real-time GPS tracking) to meet project objectives. By outlining scope and deliverables, this SOW aligns expectations and responsibilities between the client and the development team.

Scope and Objectives

The MVP will deliver two mobile apps (iOS and Android) and a responsive web portal. Key functionalities are:

- **Customer Mobile Apps (iOS/Android):** Allow users to onboard (create account and car profiles), browse services and garages, schedule service orders, choose pickup/delivery or in-person service, securely pay in-app, view order history, and access support. Core features include in-app payment integration, real-time order status updates (via notifications, photos, and chat), and basic customer support (live chat or contact form). The apps will implement UI mockups and branding per client design.
  Fleet (B2B) Dashboard: Provide corporate fleet clients with a portal to monitor all vehicles under their account in the Customer mobile apps. This includes fleet analytics (e.g. total vehicles, vehicles progress). The dashboard will support managing vehicle listings, scheduling services for multiple vehicles. Corporate users also can be a part of this. This sector will be able to see all the garages and services like the customers but get lowered price based on quantity promised per month.
- **Garage Web Portal:** Enable partner garages to log in and manage orders. Functionalities include viewing new booking requests, accepting or rejecting jobs, updating order status (e.g. "In Progress", "Completed"), uploading progress photos or notes, and communicating with customers. Garages can also manage their service offerings (adding/editing services and prices) and view payments for completed jobs. Pickup requests will be visible so garages can coordinate vehicle collection and return schedules.
- **Vehicle Pickup and Delivery:** The platform will include an option for users to request vehicle pickup from their location and return delivery after service. While live GPS tracking is excluded, the system will allow scheduling pickup and drop-off times and record locations. Pickup/delivery jobs will be managed through the order workflow.
- **Support & Communication:** Whatsapp api in app chat will connect customers with support agents. Predefined message templates (e.g. issue categories) can be provided.
- **Ratings/Feedback:** After service completion, users can rate and review the garage. Ratings will influence badges such as "Top Rated" in service listings.
- **Loyalty/Promotions:** A simple referral or loyalty system (e.g. promo codes) may be included if time allows, rewarding users for referrals or repeat usage.

Excluded (MVP): Advanced features such as live GPS vehicle tracking, automatic route optimization, and highly complex analytics are out of scope. The focus is on core booking, payment, support, and management functions, now including basic pickup/delivery scheduling and Royality programs.

This SOW covers the first phases of development and delivery. The objective is to launch a functioning MVP with the features above within the 3-month timeframe.

Functional Requirements

- **Customer Mobile App (iOS/Android):** Allow users to register (email/SMS verification) and create profiles with vehicle details. Users can browse a catalog of services and participating garages, with filtering by location, rating, and price. The booking flow lets users select a service, choose a time slot, and specify a location (either garage or home). Users can opt for vehicle pickup and return delivery, scheduling those times as part of checkout. They will see an order summary including the price (with any group discount applied) before confirming. After confirmation, the app will handle secure in-app payment (storing tokenized payment methods) and provide order status updates. Status updates (e.g. “Confirmed”, “In Progress”, “Completed”) will be delivered via push notifications and visible in the app; customers can also receive photos or messages from the garage. The app will include a request and Accpet interface for Upselling or ad-ons. After completion, users can rate and review the service. Loyalty or referral features (e.g. promo codes) can be applied if available.
- **Garage Web Portal:** Garages will have a secure login. An order dashboard will display new and pending service requests, showing customer info, requested services, and any pickup details. For each order, garage staff can accept or reject the job, and update status through stages (e.g. “In Progress”, “Completed”). They can upload progress photos or notes; each status update will notify the customer. Garages can manage their profile (contact info, working hours, services offered and pricing) and view earnings for completed jobs. They will also see scheduled pickup timing to receive the vehicle. Notifications (email or in-app alerts) will inform garages of new bookings or Requests. A simple support/helpdesk link will allow garages to contact platform support if needed via Whatsapp.
- **Fleet (B2B) Dashboard(in Customer app):** They can upload or manage their fleet vehicle list. For each vehicle, the dashboard shows service history (dates and types of services performed). The system will flag upcoming required services based on schedules or mileage, and can send alerts or notifications for maintenance. Administrators can schedule services for one or multiple fleet vehicles from the dashboard (with optional pickup if needed).analytics will display metrics such as total vehicles, progress of each vehicle.
- **Shared & Admin Features:**
- **Authentication & Security:** All users (customers, garage staff, fleet managers) will authenticate securely. Sessions will use secure tokens. Role-based access control will ensure each user sees only permitted data (e.g. a garage user cannot see other garages’ orders). Data validation and encryption will follow best practices (OWASP, PCI standards).
- **Admin Panel (Optional):** An internal admin interface may be provided for platform management. This could allow administrators to manage user accounts (approve garages, reset passwords), maintain the service catalog, view system analytics, and handle content updates. (This is optional and can be scoped based on timeline.)

Development Phases and Timeline

We will follow an Agile process over approximately 12 weeks. A representative timeline is:

- **Week 1 – Planning & Setup:** Finalize requirements and user stories (including pickup/delivery and group-buying workflows). Set up project management and backlog. Configure development environments and repositories (Git). Establish CI/CD pipelines and development standards. Review UI designs and architecture decisions with the team. Create initial data models and API design sketches.
- **Weeks 2–5 – Core Development Sprint 1:** Begin parallel development of backend and frontend prototypes. On the backend, implement basic services (user auth, data models for users, vehicles, orders, and payments) and stub out key APIs. On the mobile app, build initial screens (login, home dashboard, service listing). On the web portal, create

skeleton pages (login, dashboard layout). Integrate user registration/login (with email or SMS verification) and profile management on all platforms. Ensure basic navigation flows are functioning.

- **Weeks 6–8 – Core Development Sprint 2:** Implement the major functional components. Onboarding the booking workflow: allow selecting a service, scheduling time, and choosing pickup/ delivery options. Add the group purchase feature so multiple users can join the same booking and receive the 10% discount. Complete the garage portal order flow: displaying incoming orders, acceptance, and status updates. Integrate the payment gateway API in sandbox mode for end-toend booking/payment flow. Build the fleet dashboard features (vehicle list, upcoming services, analytics charts). Integrate chat/support functionality. Conduct weekly sprint reviews and demos; adjust priorities or designs based on feedback.
- **Weeks 9–10 – Testing & QA:** Perform thorough testing of all components.
- **Unit Testing:** Developers will write automated tests for key functions. This includes testing the booking logic (including pickup scheduling and group discount calculations), payment processing, and any complex algorithms.
- **Integration Testing:** Conduct end-to-end tests simulating typical user scenarios: booking a service (including group booking), paying, completing service, and updating status. Test third-party integrations (payment gateway, push notifications, chat) in sandbox. Verify the pickup scheduling workflow functions correctly.
- **UI/UX Testing:** Verify that the app and web UI match the design mockups. Test on various device sizes and browsers to ensure responsive layouts and smooth interactions.
- **Security Testing:** Confirm that all data is transmitted securely (HTTPS) and validate inputs to prevent SQL/NoSQL injections or other attacks. Verify authentication flows and data access controls. Ensure PCI-DSS requirements are met for payment data (using tokenization and secure endpoints).
- **User Acceptance Testing (UAT):** Engage stakeholders or a pilot user group to test the MVP flows, including registration, booking (with group purchase and pickup options), and order management. Gather feedback on usability and workflow correctness. Log any bugs or requested changes.
- **Performance Testing:** Simulate expected user load to check system responsiveness. Test API performance under typical booking volumes; optimize database queries or API calls if needed. All test results will be documented. Bugs will be tracked and prioritized for fixing. Weekly QA reports will be prepared.
- **Week 11 – Deployment Preparation:** Prepare for production deployment. Configure the production environment (set up cloud servers or services, databases, and storage). Review security certificates (SSL). Finalize any remaining test fixes. Generate release builds for iOS and Android. Submit mobile apps to the Apple App Store and Google Play Store (allowing for review time). Switch the payment gateway from sandbox to live mode (after completing KYC and approvals). Set up monitoring and logging (e.g. server health dashboards, error tracking services) to observe the live system.
- **Week 12 – Launch & Handover:** Officially launch the MVP. Conduct a final walkthrough and training session for client stakeholders (demonstrating the apps and portals). Provide documentation, including deployment steps, developer setup, and user guides (for portal/garage users and fleet managers). The system will go live for initial users.

This schedule follows industry guidelines (approximately 2–3 weeks for planning, 5–6 weeks for development, 2–3 weeks for testing/deployment). We will use agile sprints to ensure continuous progress. By focusing scope and adhering to this plan, the 3-month target is achievable.

Testing and Quality Assurance

Quality will be ensured through multiple QA activities:

- **Unit Testing:** Developers will implement automated tests for all major functions. This includes tests for booking logic (including group-buy and pickup scheduling scenarios), payment processing, user authentication, and any other critical algorithms.

- **Integration Testing:** End-to-end tests will simulate complete booking flows, including group bookings, payment transactions, and order completion. Third-party integrations (payment gateway, push notifications, chat) will be tested in their sandbox environments. Workflow from booking to garage update and user notification will be verified.
- **UI/UX Testing:** The apps and portal will be tested on various device sizes and browsers to ensure they match the provided design mockups and respond correctly. Usability testing will check that users can easily navigate through registration, booking, and payment.
- **Security Testing:** Security measures will be verified. We will ensure that all API endpoints use HTTPS, that user inputs are validated, and that authentication and authorization are enforced. Payment flows will be reviewed for PCI-DSS compliance (proper tokenization, secure storage).
- **User Acceptance Testing (UAT):** Client stakeholders or a pilot group will test key workflows to ensure the system meets business needs. This will include booking services (with and without group discounts), scheduling pickups, and managing orders in the portal. All feedback will be logged and any critical issues will be addressed before launch.
- **Performance Testing:** Basic load tests will simulate expected user activity (e.g. multiple bookings, concurrent users) to ensure the system remains responsive. Database queries and APIs will be profiled; optimizations will be applied if necessary.

All test cases and results will be documented. A final test report, including coverage and any remaining issues (with workarounds), will be delivered as part of the MVP handover.

Deployment and Maintenance

- **Deployment:** The final system will be deployed to the production cloud environment. Web portals will be hosted under secure domains with SSL certificates. The backend APIs and database will be configured for production scale. Mobile applications will be released on the App Store and Google Play Store (ready for user download). The payment gateway will be switched to live mode once all approvals are complete. Monitoring tools (e.g. logging dashboards, error tracking) will be activated to observe system health.
- **Documentation:** We will deliver comprehensive documentation. Technical documentation will cover system architecture, API endpoints, database schema, and deployment instructions. User documentation will include guides for garage administrators and fleet managers (how to use the portal, interpret the dashboard, handle orders). The code repository will include a README with setup and development instructions.
- **Post-Launch Maintenance & Support:** The development team will allocate a short support period (e.g. Three month post-launch) for bug fixes and urgent patches. Routine maintenance tasks will be defined (e.g. database backups, monitoring alerts, scheduled updates). An on-call support plan can be established for critical issues. Any limited free support hours post-launch (for tweaks or training) will be specified in the SOW agreement.

Summary

This comprehensive SOW outlines all work needed to deliver Garagat's MVP, now including vehicle pickup/ delivery scheduling and group-buying discount features. It specifies each feature and development phase, following standard SOW practices to avoid misunderstandings. By adhering to this plan and timeline, the team can deliver the mobile apps and web portals—with in-app payments, chat support, garage management, and fleet dashboard features, as well as the new pickup/delivery and group-purchase discount mechanisms—within the 3-month target. Each deliverable will meet the defined acceptance criteria to ensure a high-quality launch of Garagat's car service platform.

The technology stack (Flutter/React Native for mobile, cloud backend with a managed database, UAE payment gateway integration, etc.) follows industry best practices and the client's business requirements. All development will be documented

and carried out with quality and security in mind. This SOW provides a clear roadmap for the MVP and sets the foundation for future enhancements.
