Project Report for Frontend Team

Overview

- **[Stack]** Backend is Spring Boot 3 (java 17).
- **[Key features]** Authentication (JWT), Rides, Payments (Razorpay), Wallet crediting, Notifications (Firebase FCM), Storage (Firebase Storage).
- **[Notable config files]**

src/main/resources/application.yml

- **[Key modules]**
    - Payments:

PaymentController at /api/payments...

    - Transactions history:

PaymentTransactionController at /api/payments/history...

    - Push notifications:

PushNotificationServiceImpl (uses Firebase Admin)

    - Storage service:

FirebaseStorageServiceImpl for documents/logs

Base URLs and CORS

- **[Backend base]** http://<backend-host>:8080 (dev default)
- **[CORS]** Allowed-origins: "*" in

application.yml (app.cors.allowed-origins)

Payments (Razorpay)

- **[Create payment link]**
    - Endpoint: POST /api/payments/{rideId}
    - Auth: Authorization: Bearer <JWT>
    - Response: PaymentLinkResponse with paymentLinkUrl, paymentLinkId
    - Notes populated: ride_id, driver_id, user_id
    - Callback URL
      embedded: http://localhost:3000/ride/{rideId}/payment/success (change to prod domain)
- **[Redirect callback]** (used by Razorpay Payment Link)
    - Endpoint: GET /api/payments
    - Query params: payment_id, order_id (rideId)

        - Behavior: Verifies capture and updates Ride.paymentDetails.paymentStatus
        - Frontend: Implement the next UI after success (e.g., payment success page)
- **[Webhook]** (server-to-server)
    - Endpoint: POST /api/webhooks/razorpay
    - Header: X-Razorpay-Signature verified
    - Behavior:
        - Parses ride_id from payload notes
        - Marks payment as COMPLETED on Ride
        - Credits driver wallet (fare minus commission)
        - Sends push notification to the rider (details below)
    - Frontend: No action required, but UI should reflect status transitions (e.g., polling/WS if needed)

Payment History APIs

- **[User history]** GET /api/payments/history/user
  Header: Authorization: Bearer <JWT>
- **[Driver history]** GET /api/payments/history/driver
  Header: Authorization: Bearer <JWT>
- **[Ride history]** GET /api/payments/history/ride/{rideId}

Note: The model

PaymentTransaction exists, and history endpoints are present, but creation of transaction records on payment success is pending. Frontend can still consume these endpoints if/when data is populated.

Push Notifications (Firebase FCM)

- **[Implementation]**

PushNotificationServiceImpl uses FirebaseMessaging (Admin SDK).

- **[Trigger]** On Razorpay webhook success, backend sends a push to the rider:
    - Title: “Payment received”
    - Body: “Your ride payment was successful”
    - Data: event=payment_success, ride_id=<id>
- **[Frontend expectations]**
    - Obtain FCM token via client SDK.
    - Provide token to backend (store in MyUser.fcmToken). If there isn’t an existing endpoint, we will add one; expect POST /api/users/me/fcm-token with { token: string }.

        - Handle push notifications in-app:
            - Foreground: show in-app banner/modal.
            - Background: navigate using data.ride_id when notification is tapped.
- **[Topics]** Not used; per-token deliveries now.

Firebase Storage (Documents & Logs)

- **[Service]**

FirebaseStorageServiceImpl

- **[Buckets/paths]**
    - Driver documents: gs://gauva-15d9a.firebasestorage.app/documents
    - Logs: gs://gauva-15d9a.firebasestorage.app/app_logs
- **[Size limits]**
    - Documents/photos: max 2 MB
    - Logs: max 5 MB
- **[Frontend expectations]**
    - Upload endpoints for driver KYC/profile photos will accept multipart/form-data with Content-Type: image/jpeg|png or pdf if allowed.
    - Returned value: a gs://... path; if HTTP URL is needed for display, we will add signed URL or use Firebase client SDK for access.
    - Until endpoints are added, frontend can prepare UI with file size validation (2 MB docs/photos) and content type checks.

Environment Variables the Frontend Should Know About

- These affect behavior the frontend relies on:
    - FIREBASE_PROJECT_ID, FIREBASE_STORAGE_BUCKET (for Firebase SDK config on web/app)
    - Razorpay test vs production keys are held server-side, but frontend should use the returned paymentLinkUrl
    - Callback URL in

PaymentController should be set to the real frontend domain (currently localhost)

Data Contracts and Models

- **[Ride]** Ride includes id, fare, user, driver, paymentDetails
- **[PaymentDetails]** paymentId, paymentStatus where PaymentStatus can be COMPLETED
- **[User]**

MyUser includes id, email, phone, fullName, fcmToken, profileImage

- **[Driver]**

Driver includes id, name, email, mobile

- **[Transactions]**

PaymentTransaction provides structured history with provider, type, status, amount, currency, refs

Integration Flows

- **[Payment via Link]**
    1. Frontend calls POST /api/payments/{rideId} with JWT.
    2. Backend returns paymentLinkUrl.
    3. Frontend sends user to paymentLinkUrl.
    4. Razorpay redirects to frontend success page (callback URL).
    5. Backend webhook finalizes payment and triggers push; UI should reflect finalized state.
- **[Payment via QR]**
    - Endpoint: POST /api/payments/{rideId}/qr
    - Response includes qr_id, image_url, amount.
    - Frontend can display the QR image_url for offline UPI payment; webhook confirms payment.
- **[Push Handling]**
    - On app load/login, obtain FCM token; send to backend.
    - Listen for push event and navigate accordingly (payment_success → ride detail screen using ride_id).
- **[Document Upload]**
    - Frontend selects file (validate ≤ 2MB).
    - Send multipart/form-data to the upcoming endpoint.
    - Display upload progress and success/failure; show image preview/URL if returned.

Error Handling Expectations

- **[Payments]** Show user-friendly message if link creation fails. Offer retry.
- **[Webhook]** Frontend does not interact directly; may poll ride status if needed.
- **[Uploads]** Handle 400 errors on validation (oversize/unsupported type).

Security Notes

- Razorpay keys and webhook secrets are server-side.
- Firebase service account is server-only; do not bundle into the frontend.

- Enforce content types and size limits client-side to reduce failures.
- Prefer private document access; if public display is needed, we'll add signed URL generation.

What's Pending for Frontend Collaboration

- **[FCM token endpoint]** We'll add POST /api/users/me/fcm-token to store device tokens.
- **[Upload endpoints]** We'll add driver document/profile image endpoints with content-type checks and return payload format.
- **[Callback URL]** We will switch hardcoded callback URL to your production domain; please share the URL.
- **[Transaction logging]** We'll persist

PaymentTransaction on webhook to improve the history pages.

References to Code

- PaymentController at

src/main/java/com/ridefast/ride_fast_backend/controller/PaymentController.java

- PaymentTransactionController at

src/main/java/com/ridefast/ride_fast_backend/controller/PaymentTransactionController.java

- PushNotificationServiceImpl at

src/main/java/com/ridefast/ride_fast_backend/service/notification/impl/PushNotificationServiceImpl.java

- FirebaseStorageServiceImpl at

src/main/java/com/ridefast/ride_fast_backend/service/storage/impl/FirebaseStorageServiceImpl.java

- application.yml at

src/main/resources/application.yml (see app.firebase.*, app.razorpay.*)

Requested Inputs from Frontend

- **[Production frontend URL]** for Razorpay callback_url.
- **[FCM integration]** Confirm platform(s) using FCM and token lifecycle (refresh cadence).
- **[Upload UX]** Confirm file types allowed for documents/photos and any cropping or compression you'll apply client-side.

Recommended Actions

- **[Provide production callback URL]** so we can update

PaymentController.

- **[Agree on endpoints]** for:
    - Saving FCM token (POST /api/users/me/fcm-token)

    - Upload driver docs/photos (endpoint paths and request fields)
- **[Align on notification UX]** for payment success flow and deep-links.

Status

- Backend payments, webhook, and push to rider are working; more wiring can be added per your UX.
- Firebase Storage ready with size limits; awaiting upload endpoints design confirmation.

Feedback submitted