

BANXX — API Specification

37 endpoints · UAT · 8 October 2026 · app v1.8.2

Every HTTP call the BANXX app (banxx-app_V1) makes, transcribed from the app source. Each of the 37 endpoints has a live caller in the app, and every page names it under “Called from”. Endpoints that are coded in the app but that no screen calls are left out.

Revision

Supersedes the 8 September 2026 revision (35 endpoints, app v1.6.5). It adds the Logout API, the three business loan calls and the legal pages call, encrypts the sensitive request bodies, and moves the partner APIs onto the BANXX host. Section 0 lists every difference.

Base URLs

Constant	Base URL	What lives there
banxxBaseUrl	https://uat.banxx.com:8000/in/home/app/api	Auth and session (login, OTP, refresh, logout), profile, notifications, banners, vouchers, product list, the personal / gold / housing partner APIs, offer summary, lender hand-off, legal pages
chatBaseUrl	https://uat.banxx.com:8000/in/home/app/api/chat/	Charles — AI chat, voice command, AI advice, FAQ and support chat
businessLoanBaseUrl	https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth	Business loan partner APIs — create lead, update lead, offers
banxxWebBaseUrl	https://uat.banxx.com/in/home/app	Not an API. The lender journeys the hand-off calls redirect into, opened in a web view. Same host as banxxBaseUrl but on the default port — these paths 404 on :8000.

0. What changed since the 8 September 2026 revision

This revision brings the document in line with app v1.8.2: one new session API (Logout), the business loan and legal pages calls, encrypted request bodies, the BANXX host for the partner calls, and the removal of endpoints that no screen calls.

0.1 Logout API added — POST /logout

The app now tells the server when a session ends, so the token is invalidated server-side and a request replayed with it afterwards is rejected. Before this, logging out only cleared local data and the token stayed valid until it expired. The app calls it from the Logout buttons, on a forced logout, after account deletion, and when the app is closed (reason "App closed"). The local wipe runs whether or not the call succeeds. See section 4.

0.2 Refresh token travels with the session

/verifyotp now returns refresh_token next to token. /refresh-token sends it back in the body as {"refresh_token": "..."} (the current token is still the Bearer), and its reply can carry a rotated refresh_token. The refresh outcome is read from `status` as the string "true".

0.3 Sensitive payloads are encrypted

Request bodies on these routes are sent as {"data": "<envelope>"} (AES-256-GCM body, key wrapped with RSA-OAEP) and the reply is decrypted by the app: /send-otp, /verifyotp, /resend-otp (an encrypted empty object), POST /v2/personal-data/{credit_uid}, and every POST under /partner-auth/ (personal, gold, housing and business loans).

GET routes have no body, so the server encrypts the reply itself: GET /v2/personal-data/*, GET /user-mobile/* and every GET under /partner-auth/ come back as {"data": "<envelope>"}. Every page shows an Encryption row where this applies, and the JSON on the page is always the plain, decrypted form. Everything else is plain JSON. The encrypt and decrypt steps are in section 16.

0.4 Paths and hosts that moved

Endpoint	Was	Is now
POST /partner-auth/dedupe	https://loannet.in:8000/api	https://uat.banxx.com:8000/in/home/app/api
POST /v2/partner-auth/create-lead	https://loannet.in:8000/api	https://uat.banxx.com:8000/in/home/app/api
POST /v2/partner-auth/update-lead/{lead_id}	https://loannet.in:8000/api	https://uat.banxx.com:8000/in/home/app/api
GET /partner-auth/get-offers/{lead_id}	https://loannet.in:8000/api	https://uat.banxx.com:8000/in/home/app/api
POST /v2/partner-auth/gold-loans	https://loannet.in:8000/api	https://uat.banxx.com:8000/in/home/app/api
POST /v2/partner-auth/housing-loan	https://loannet.in:8000/api	https://uat.banxx.com:8000/in/home/app/api
User record	GET /user/{credit_uid}	GET /user-mobile/{credit_uid}

0.5 Delete account sends no body

POST /banxx-app/delete-account no longer sends the phone number. The server deletes the account the bearer token belongs to.

0.6 New in this revision

Endpoint	What it does	Section
POST /logout	Ends the session on the server	4
POST /v2/partner-auth/bl-create-lead	Creates a business loan lead	6
POST /v2/partner-auth/bl-update-lead/{lead_id}	Updates a business loan lead	6
GET /v2/partner-auth/bl-get-offers/{lead_id}	Business loan offers	6
GET /legal/{slug}	Six legal pages	15

Index

#	Section	Endpoints	Base
0	What changed	Revision notes	—
1	Login	POST /send-otp	banxxBaseUrl
2	OTP Verification	POST /verifyotp POST /resend-otp/{app_id}	banxxBaseUrl
3	Signup — Personal Details Registration	POST /v2/personal-data/{credit_uid}	banxxBaseUrl
4	Session management	POST /refresh-token POST /logout	banxxBaseUrl
5	Dashboard	GET /v2/personal-data/{credit_uid} GET /user-mobile/{credit_uid} GET /product-list	banxxBaseUrl
6	Offers and Credit — partner APIs	POST /partner-auth/dedupe POST /v2/partner-auth/create-lead POST /v2/partner-auth/update-lead/{lead_id} GET /partner-auth/get-offers/{lead_id} POST /v2/partner-auth/gold-loans POST /v2/partner-auth/housing-loan POST /v2/partner-auth/bl-create-lead POST /v2/partner-auth/bl-update-lead/{lead_id} GET /v2/partner-auth/bl-get-offers/{lead_id}	banxxBaseUrl (bl-*, businessLoanBaseUrl)
7	Loan status and lender hand-off	GET /partner-auth/get-summary/{uuid} POST /gold-loans/{creditapp_uuid} POST /housing-loan/{creditapp_uuid}	banxxBaseUrl
8	Notifications	GET /banxx/notifications GET /banxx/notifications/fetch/{id}	banxxBaseUrl
9	Promotional banners	GET /banxx/banners	banxxBaseUrl
10	AI Chat and Voice Command	POST chat/start POST chat/message GET chat/questions	chatBaseUrl
11	AI Advice	POST chat/advice	chatBaseUrl
12	Deals and Offers (vouchers)	POST /vouchagram/getbrands	banxxBaseUrl
13	Support — Help and Chat	GET chat/faq POST chat/faq-feedback GET chat/conversations GET chat/history/{conversation_id}	chatBaseUrl
14	Settings	GET /banxx/notification-preferences POST /banxx/notification-preferences POST /banxx-app/delete-account	banxxBaseUrl
15	Legal and compliance pages	GET /legal/{slug}	banxxBaseUrl
16	Payload encryption	How to encrypt and decrypt	—

The Base column names the constant; its URL is on page 1.

1. Login

Base URL: `https://uat.banxx.com:8000/in/home/app/api`

Send OTP

Endpoint	POST /send-otp
Full URL	<code>https://uat.banxx.com:8000/in/home/app/api/send-otp</code>
Called from	Login screen — BanxxAuthService.sendOtp()
Request headers	Content-Type: application/json Accept: application/json
Encryption	Request body is encrypted. The app sends {"data": "<envelope>"} (AES-256-GCM body, key wrapped with RSA-OAEP). The JSON shown on this page is what the app builds before encrypting it, and what the server reads after decrypting it. A reply that comes back as {"data": "<envelope>"} is decrypted by the app; the response shown is the decrypted JSON.
Request body	mobile_phone_number required The 10-digit mobile number. Sent as a JSON number when it parses as one, otherwise as a string. <pre>{ "mobile_phone_number": 9876543210 }</pre>
Response body	<pre>{ "success": "true", "app_id": "a91f4c7e-2b18-4d55-9f30-6c1e8ab47d02", "message": "OTP sent successfully" }</pre> success is read as a string or a boolean. app_id is carried into /verifyotp and /resend-otp.
Errors	The message shown to the user is taken from `message`, or from the first entry of a Laravel `errors` map when `message` is a generic validation sentence. Fallback: "Failed to send OTP. Please try again."
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/send-otp' \ -H 'Content-Type: application/json' \ -H 'Accept: application/json' \ -d '{"mobile_phone_number": 9876543210}'</pre>

2. OTP Verification

Base URL: <https://uat.banxx.com:8000/in/home/app/api>

Verify OTP

Endpoint	POST /verifyotp
Full URL	https://uat.banxx.com:8000/in/home/app/api/verifyotp
Called from	OTP screen — BanxxAuthService.verifyOtp()
Request headers	Content-Type: application/json
Encryption	Request body is encrypted. The app sends {"data": "<envelope>"} (AES-256-GCM body, key wrapped with RSA-OAEP). The JSON shown on this page is what the app builds before encrypting it, and what the server reads after decrypting it. A reply that comes back as {"data": "<envelope>"} is decrypted by the app; the response shown is the decrypted JSON.
Request body	code required The OTP. Sent as a JSON number when it parses as one. credit_prospect_id required The app_id from /send-otp. { "code": 123456, "credit_prospect_id": "a91f4c7e-2b18-4d55-9f30-6c1e8ab47d02" }
Response body	{ "success": "true", "message": "OTP verified", "token": "eyJ0eXAiOiJKV1QiLCJhbGciOi...", "refresh_token": "def50200a1b2c3d4...", "greeting": "Hi Rahul", "lastLoginDateTime": "2026-08-19T06:42:11Z", "data": { "credituid": "cp_8f21d0b4e7a3", "greeting": "Hi Rahul", "lastLoginDateTime": "2026-08-19T06:42:11Z" } } token and data.credituid are written to secure storage and key every authenticated call thereafter. refresh_token is stored too and is what /refresh-token and the session depend on (read from the root or from data). greeting and lastLoginDateTime are read from the root or from data, whichever carries them.
cURL	curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/verifyotp' \ -H 'Content-Type: application/json' \ -d '{"code": 123456, "credit_prospect_id": "a91f...d02"}'

Resend OTP

Endpoint	POST /resend-otp/{app_id}
Full URL	https://uat.banxx.com:8000/in/home/app/api/resend-otp/{app_id}
Called from	OTP screen, Resend link — BanxxAuthService.resendOtp(). The link is locked for 75 seconds after each send.
Request headers	Content-Type: application/json
Encryption	The app sends an encrypted empty object ({"data": "<envelope>"}) so the server answers encrypted too. The response shown is the decrypted JSON.
Path parameters	app_id required The login attempt id returned by /send-otp.
Request body	None.
Response body	{ "success": "true", "app_id": "a91f4c7e-2b18-4d55-9f30-6c1e8ab47d02", "message": "OTP resent successfully" } A new app_id in this response replaces the one held by the screen.
cURL	curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/resend-otp/{app_id}' \ -H 'Content-Type: application/json'

3. Signup — Personal Details Registration

Base URL: `https://uat.banxx.com:8000/in/home/app/api`

Save personal data

Endpoint	POST /v2/personal-data/{credit_uid}
Full URL	<code>https://uat.banxx.com:8000/in/home/app/api/v2/personal-data/{credit_uid}</code>
Called from	Registration screen and Settings > My Profile — <code>BanxxAuthService.savePersonalData()</code>
Request headers	Authorization: Bearer <token> Content-Type: application/json
Encryption	Request body is encrypted. The app sends <code>{"data": "<envelope>"}</code> (AES-256-GCM body, key wrapped with RSA-OAEP). The JSON shown on this page is what the app builds before encrypting it, and what the server reads after decrypting it. A reply that comes back as <code>{"data": "<envelope>"}</code> is decrypted by the app; the response shown is the decrypted JSON.
Path parameters	<code>credit_uid</code> required The credit prospect uid.
Request body	<code>first_name</code> required <code>last_name</code> required <code>email</code> required <code>birth_date</code> required ISO date, YYYY-MM-DD. The app enforces an age of 14-100 before sending. <pre>{ "first_name": "Rahul", "last_name": "Sharma", "email": "rahul.sharma@example.com", "birth_date": "1992-04-17" }</pre>
Response body	<pre>{ "success": "true", "message": "Details saved" }</pre> HTTP 200 with <code>success:false</code> is treated as a failure and <code>`message`</code> is shown to the user — that is how "The email has already been taken." reaches the screen. A success drops the app's cached profile so every screen re-reads the new values.
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/v2/personal-data/cp_8f21d0b4e7a3' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{"first_name": "Rahul", "last_name": "Sharma", "email": "rahul.sharma@example.com", "birth_date": "1992-04-17"}'</pre>

4. Session management

Base URL: `https://uat.banxx.com:8000/in/home/app/api`

Refresh token

Endpoint	POST /refresh-token
Full URL	<code>https://uat.banxx.com:8000/in/home/app/api/refresh-token</code>
Called from	Banxx401RefreshInterceptor, via TokenRefreshCoordinator — fired automatically on any 401 from a bearer-authenticated endpoint, and on a timer before the token expires. The failed request is retried once with the new token.
Request headers	Authorization: Bearer <current token> X-CSRF-TOKEN: (sent as an empty value)
Request body	refresh_token required The refresh token returned by /verifyotp (or by the last refresh). <pre>{ "refresh_token": "def50200a1b2c3d4..." }</pre>
Response body	<pre>{ "status": "true", "token": "eyJ0eXAiOiJKV1QiLCJhbGciOi...", "refresh_token": "def50200e5f6a7b8...", "message": "Token refreshed" }</pre> The outcome is read from `status` (or `success`), sent as the string "true" — a 1 or true is accepted too. refresh_token is a rotated refresh token and is optional: when it is absent the stored one stays valid. An empty or failed response signs the user out. This call goes out on a bare client with no interceptor, so a 401 on the refresh itself never recurses.
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/refresh-token' \ -H 'Authorization: Bearer <token>' \ -H 'X-CSRF-TOKEN;' \ -H 'Content-Type: application/json' \ -d '{"refresh_token": "def50200a1b2c3d4..."}'</pre>

Logout

Endpoint	POST /logout
Full URL	<code>https://uat.banxx.com:8000/in/home/app/api/logout</code>
Called from	Every way out of a session, through BanxxAuthService.logout(): the Logout buttons (reason "User requested logout"), the forced logout when the refresh token is dead, account deletion, and when the app is closed (reason "App closed", fired on the next launch if the process was killed). The call is best-effort: it never blocks or fails the logout, and the local wipe of secure storage and caches runs whether or not it succeeds.
Request headers	Authorization: Bearer <token> X-CSRF-TOKEN: (sent as an empty value) Accept: application/json Content-Type: application/json
Request body	logout_reason required Why the session ended — "User requested logout" or "App closed". Lets the server tell the Logout button apart from an app close. <pre>{ "logout_reason": "User requested logout" }</pre>
Response body	Not read by the app — any 2xx counts as logged out. On the server this must invalidate the token, so a request replayed with it afterwards is rejected with a 401 (PT finding: weak session management / session not invalidated on close). A 401 here means the token is already dead, which is the outcome wanted. Short 5-second connect and receive timeouts apply, because the user is waiting on the logout screen. The call goes out on a bare client with no interceptor, so it never triggers a refresh.
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/logout' \ -H 'Authorization: Bearer <token>' \ -H 'X-CSRF-TOKEN;' \ -H 'Accept: application/json' \</pre>

	<pre>-H 'Content-Type: application/json' \ -d '{"logout_reason": "User requested logout"}'</pre>
--	--

5. Dashboard

Base URL: <https://uat.banxx.com:8000/in/home/app/api>

Get personal data

Endpoint	GET /v2/personal-data/{credit_uid}
Full URL	https://uat.banxx.com:8000/in/home/app/api/v2/personal-data/{credit_uid}
Called from	Prefetched at launch and after OTP verification, then shared from an in-memory cache by the dashboard, Charles strip, nav header, Settings, Activity, the offers bloc and the lead form — BanxxAuthService.getPersonalData()
Request headers	Authorization: Bearer <token>
Encryption	No request body. The server always answers this route encrypted, as {"data": "<envelope>"}; the app decrypts it. The response shown on this page is the decrypted JSON.
Path parameters	credit_uid required The credit prospect uid returned as data.credituid by /verifyotp.
Response body	<pre>{ "data": { "first_name": "Rahul", "last_name": "Sharma", "email": "rahul.sharma@example.com", "birth_date": "1992-04-17", "creditprospect_uuid": "cp_8f21d0b4e7a3", "messageTitle": "3 offers are waiting for you", "messageBody": "Two personal loans and a gold loan matched your profile." } }</pre> The same fields are also accepted at the root. messageTitle and messageBody are the authoritative copy for the Charles strip. A user with no profile yet returns HTTP 200 with an empty body — that is "not onboarded", not a failure, and is not cached. This is the call whose answer decides whether the OTP screen goes to signup or straight to the dashboard.
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/v2/personal-data/cp_8f21d0b4e7a3' -H 'Authorization: Bearer <token>'</pre>

Get user record

Endpoint	GET /user-mobile/{credit_uid}
Full URL	https://uat.banxx.com:8000/in/home/app/api/user-mobile/{credit_uid}
Called from	Prefetched at launch; read by the offers bloc, the Applications tab and the loan status tracker — BanxxAuthService.getUserDetails() and LoanApplicationService.fetchUserRecord()
Request headers	Authorization: Bearer <token> X-CSRF-TOKEN: (sent as an empty value by the loan-status caller)
Encryption	No request body. The server always answers this route encrypted, as {"data": "<envelope>"}; the app decrypts it. The response shown on this page is the decrypted JSON.
Path parameters	credit_uid required The creditprospect_uuid.
Response body	<pre>{ "data": { "creditapp_uuid": "ca_5b7290ffe14c", "creditprospect_uuid": "cp_8f21d0b4e7a3", "pl_completed": 1, "gl_completed": 0, "hl_completed": 1, "tin": "ABCDE1234F", "pincode": "400072", "monthly_income": "85000", "picked_lenders": "Muthoot,IIFL", "knockout_lenders": "HDFC", "updated_at": "2026-08-20T11:04:52Z" } }</pre>

	<pre> } } </pre> Never cached across calls: the *_completed flags change the moment an application is submitted. creditapp_uuid is what keys the two lender hand-off calls, so this request always runs before them. Flags are accepted as 1, "1" or true.
cURL	<pre> curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/user-mobile/cp_8f21d0b4e7a3' \ -H 'Authorization: Bearer <token>' </pre>

Product list

Endpoint	GET /product-list
Full URL	https://uat.banxx.com:8000/in/home/app/api/product-list
Called from	Credit-card offers card, insurance card and the product selection screens — ProductListApiService
Request headers	Authorization: Bearer <token> Accept: application/json
Response body	<pre> { "data": [{ "category_name": "Insurance", "long_description": "Cover for you and your family", "short_description": "Insurance", "category_img_thumb": "https://cdn.banxx.com/cat/insurance.png", "subcategories": [{ "category_name": "Health Insurance", "long_description": "Cashless cover at 7,000+ hospitals", "subcategories": [{ "category_name": "FlashAid", "utm_link": "https://partner.example.com/apply" }] }] }] } </pre> A bare top-level array is also accepted. Cached for the session; a failed fetch is remembered so a card can tell an API failure apart from a genuinely empty catalogue. Insurance is located by a category_name containing "insurance"; a subcategory with no children is treated as the offer itself.
cURL	<pre> curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/product-list' \ -H 'Authorization: Bearer <token>' \ -H 'Accept: application/json' </pre>

6. Offers and Credit — partner APIs

Base URL: <https://uat.banxx.com:8000/in/home/app/api>

These calls are served from the BANXX host with the bearer token. The personal, gold and housing loan calls are below; the three business loan calls follow them. The server rejects a mobileNumber that does not belong to the token's user with a 403.

Dedupe check

Endpoint	POST /partner-auth/dedupe
Full URL	https://uat.banxx.com:8000/in/home/app/api/partner-auth/dedupe
Called from	Offers bloc, before the lead form is offered — OffersRemoteDataSource.checkDedupe()
Request headers	Authorization: Bearer <token> Content-Type: application/json
Encryption	Request body is encrypted. The app sends {"data": "<envelope>"} (AES-256-GCM body, key wrapped with RSA-OAEP). The JSON shown on this page is what the app builds before encrypting it, and what the server reads after decrypting it. A reply that comes back as {"data": "<envelope>"} is decrypted by the app; the response shown is the decrypted JSON.
Request body	mobileNumber required string { "mobileNumber": "9876543210" }
Response body	{ "success": "true", "leadId": "LN2026081900412" } When no lead exists, success is not "true" and no leadId is returned — the app then shows the lead creation form.
cURL	curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/partner-auth/dedupe' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{"mobileNumber": "9876543210"}'

Create lead

Endpoint	POST /v2/partner-auth/create-lead
Full URL	https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/create-lead
Called from	Personal loan lead form — OffersRemoteDataSource.createLead()
Request headers	Authorization: Bearer <token> Content-Type: application/json
Encryption	Request body is encrypted. The app sends {"data": "<envelope>"} (AES-256-GCM body, key wrapped with RSA-OAEP). The JSON shown on this page is what the app builds before encrypting it, and what the server reads after decrypting it. A reply that comes back as {"data": "<envelope>"} is decrypted by the app; the response shown is the decrypted JSON.
Request body	mobileNumber required string firstName required string lastName required string pan required string dob required YYYY-MM-DD email required string pincode required string monthlyIncome required number employmentStatus required number consumerConsentDate required built by the app consumerConsentIp required built by the app employerName optional sent when employmentStatus is 1 officePincode optional sent when employmentStatus is 1 businessRegistrationType optional number

	residenceType optional number businessCurrentTurnover optional number businessYears optional number businessAccount optional number { "mobileNumber": "9876543210", "firstName": "Rahul", "lastName": "Sharma", "pan": "ABCDE1234F", "dob": "1992-04-17", "email": "rahul.sharma@example.com", "pincode": "400072", "monthlyIncome": 85000, "employmentStatus": 1, "consumerConsentDate": "2026-08-21 10:15:00", "consumerConsentIp": "49.36.180.14" }
Response body	{ "success": "true", "leadId": "LN2026082100518" } "id" is accepted in place of "leadId". A refusal comes back HTTP 200 with no lead id and the reason in the body; the app raises that reason verbatim rather than reporting a generic failure.
cURL	curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/create-lead' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{ ...the lead payload above... }'

Update lead

Endpoint	POST /v2/partner-auth/update-lead/{lead_id}
Full URL	https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/update-lead/{lead_id}
Called from	Lead form, when a lead already exists — OffersRemoteDataSource.updateLead()
Request headers	Authorization: Bearer <token> Content-Type: application/json
Encryption	Request body is encrypted. The app sends {"data": "<envelope>"} (AES-256-GCM body, key wrapped with RSA-OAEP). The JSON shown on this page is what the app builds before encrypting it, and what the server reads after decrypting it. A reply that comes back as {"data": "<envelope>"} is decrypted by the app; the response shown is the decrypted JSON.
Path parameters	lead_id required Taken from the event when supplied, otherwise the lead id in local storage. If neither is available the call is not made and the screen shows "No lead ID found. Please create a lead first."
Request body	The complete lead payload — the same fields as create-lead, not only the changed ones. mobileNumber required string firstName required string lastName required string pan required string dob required YYYY-MM-DD email required string pincode required string monthlyIncome required number employmentStatus required number consumerConsentDate required built by the app consumerConsentIp required built by the app employerName optional sent when employmentStatus is 1 officePincode optional sent when employmentStatus is 1 businessRegistrationType optional number

	residenceType optional number businessCurrentTurnover optional number businessYears optional number businessAccount optional number
Response body	Not read by the app.
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/update-lead/LN2026082100518' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{ ...the full lead payload... }'</pre>

Get offers

Endpoint	GET /partner-auth/get-offers/{lead_id}
Full URL	https://uat.banxx.com:8000/in/home/app/api/partner-auth/get-offers/{lead_id}
Called from	Personal / gold / housing loan offer carousels — OffersRemoteDataSource.getOffers()
Request headers	Authorization: Bearer <token> Content-Type: application/json
Encryption	No request body. The server always answers this route encrypted, as {"data": "<envelope>"}; the app decrypts it. The response shown on this page is the decrypted JSON.
Path parameters	lead_id required The lead id, or the creditprospect_uuid when the personal loan is already completed.
Response body	<pre>{ "success": "true", "leadId": "LN2026082100518", "offers": [{ "lenderId": 12, "lenderName": "Muthoot Finance", "lenderLogo": "https://cdn.loannet.in/lenders/muthoot.png", "lenderLogoDark": ".../muthoot-dark.png", "description": "Gold loan at attractive rates", "offerAmountUpTo": "500000", "offerTenure": "12 months", "offerInterestRate": "12.5%", "offerProcessingFees": "1%", "offerLink": "https://partner.example.com/apply/690289", "kfs": "https://cdn.loannet.in/kfs/muthoot-gold.pdf", "status": "Accepted", "lenderLeadId": "MU-88213", "lenderLeadType": "gold" }], "unmatchedOffers": [] }</pre> offers[] renders the cards; unmatchedOffers[] renders the "not matched" list. Field names are accepted in both spellings (lenderLogo / LenderLogo, lenderId / LenderId). The KFS link is read from kfs, Kfs, KFS, kfsUrl, kfsLink or keyFactStatement, and offerLink / applyUrl / redirectionUrl all work as the apply URL.
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/partner-auth/get-offers/LN2026082100518' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json'</pre>

Gold loan application

Endpoint	POST /v2/partner-auth/gold-loans
Full URL	https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/gold-loans
Called from	Gold loan form — OffersRemoteDataSource.createGoldLoan()

Request headers	Authorization: Bearer <token> Content-Type: application/json
Encryption	Request body is encrypted. The app sends {"data": "<envelope>"} (AES-256-GCM body, key wrapped with RSA-OAEP). The JSON shown on this page is what the app builds before encrypting it, and what the server reads after decrypting it. A reply that comes back as {"data": "<envelope>"} is decrypted by the app; the response shown is the decrypted JSON.
Request body	mobileNumber required string firstName required string lastName required string pan required string email required string pincode required string loanAmount required number consumerConsentDate required built by the app consumerConsentIp required built by the app { "mobileNumber": "9876543210", "firstName": "Rahul", "lastName": "Sharma", "pan": "ABCDE1234F", "email": "rahul.sharma@example.com", "pincode": "400072", "loanAmount": 250000, "consumerConsentDate": "2026-08-21 10:15:00", "consumerConsentIp": "49.36.180.14" }
Response body	The same structure as /partner-auth/get-offers — success, message, offers[], unmatchedOffers[] — carrying the lenders matched for this application and their opening status. Each offer carries its own opening `status`. A rejection answers HTTP 200 with success:false and no offers; the app raises `message` so the user is told why instead of being shown an empty card. Only the opening status per lender is read; this build makes no second call for the later ones.
cURL	curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/gold-loans' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{ ...the gold payload above... }'

Housing loan application

Endpoint	POST /v2/partner-auth/housing-loan
Full URL	https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/housing-loan
Called from	Housing loan form — OffersRemoteDataSource.createHousingLoan()
Request headers	Authorization: Bearer <token> Content-Type: application/json
Encryption	Request body is encrypted. The app sends {"data": "<envelope>"} (AES-256-GCM body, key wrapped with RSA-OAEP). The JSON shown on this page is what the app builds before encrypting it, and what the server reads after decrypting it. A reply that comes back as {"data": "<envelope>"} is decrypted by the app; the response shown is the decrypted JSON.
Request body	mobileNumber required string firstName required string lastName required string pan required string dob required YYYY-MM-DD email required string pincode required string monthlyIncome required number housingLoanAmount required number

	propertyType required string — House Flat Plot consumerConsentDate required built by the app consumerConsentIp required built by the app <pre>{ "mobileNumber": "9876543210", "firstName": "Rahul", "lastName": "Sharma", "pan": "ABCDE1234F", "dob": "1992-04-17", "email": "rahul.sharma@example.com", "pincode": "400072", "monthlyIncome": 85000, "housingLoanAmount": 4500000, "propertyType": "House", "consumerConsentDate": "2026-08-21 10:15:00", "consumerConsentIp": "49.36.180.14" }</pre>
Response body	The same structure as /partner-auth/get-offers and the gold loan call, and the same HTTP 200 rejection handling.
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/housing-loan' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{ ...the housing payload above... }'</pre>

Business loan calls (bl-create-lead, bl-update-lead, bl-get-offers) use businessLoanBaseUrl, which is <https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth>. The bl-dedupe and bl-get-summary routes exist in the app's data layer but no screen calls them, so they are not documented here.

Create business lead

Endpoint	POST /v2/partner-auth/bl-create-lead		
Full URL	https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/bl-create-lead		
Called from	Business Loan lead form — OffersRemoteDataSource.createBusinessLead(), through CreateBusinessLeadEvent. A business lead has its own leadId, stored separately from the personal one, so a user can hold both.		
Request headers	Authorization: Bearer <token> Content-Type: application/json		
Encryption	Request body is encrypted. The app sends {"data": "<envelope>"} (AES-256-GCM body, key wrapped with RSA-OAEP). The JSON shown on this page is what the app builds before encrypting it, and what the server reads after decrypting it. A reply that comes back as {"data": "<envelope>"} is decrypted by the app; the response shown is the decrypted JSON.		
Request body	mobileNumber	required	string
	firstName	required	string
	lastName	required	string
	pan	required	string
	dob	required	YYYY-MM-DD
	email	required	string
	pincode	required	string
	monthlyIncome	required	number
	businessRegistrationType	required	number, 1-7 (see codes below)
	residenceType	required	number, 1-2
	businessEntityType	required	number, 1-4
	businessCurrentTurnover	required	number, 2-4
	businessYears	required	number, 1-3
	businessAccount	required	number, 1-2
	consumerConsentDate	required	built by the app, YYYY-MM-DD HH:MM:SS
	consumerConsentIp	required	built by the app
	creditScoreClass	optional	number, 1-3
	glIntent	optional	number, 1-2
	waitForAllOffers	optional	number, 1-2
	Codes. businessRegistrationType: 1 GST, 2 Shop & establishment, 3 Municipal corporation, 4 Palika / Gramapanchayat, 5 Udyog Aadhar, 6 Drugs licence / food and drugs control certificate, 7 Other. residenceType: 1 Rented, 2 Owned. businessEntityType: 1 Proprietorship, 2 Partnership, 3 Pvt Ltd, 4 LLP.		

	businessCurrentTurnover: 2 up to 12 lacs, 3 12-30 lacs, 4 above 30 lacs. businessYears: 1 under 1 year, 2 1-2 years, 3 over 2 years. businessAccount (current account in the business name): 1 Yes, 2 No. creditScoreClass: 1 Prime, 2 Near Prime, 3 Sub-Prime. glIntent (also wants gold loan offers): 1 Yes, 2 No. waitForAllOffers: 1 wait for every offer (180 s max), 2 do not wait; left unset the API waits 15 s. <pre>{ "mobileNumber": "9876543210", "firstName": "Rahul", "lastName": "Sharma", "pan": "ABCDE1234F", "dob": "1992-04-17", "email": "rahul.sharma@example.com", "pincode": "400072", "monthlyIncome": 85000, "businessRegistrationType": 1, "residenceType": 2, "businessEntityType": 3, "businessCurrentTurnover": 3, "businessYears": 3, "businessAccount": 1, "consumerConsentDate": "2026-08-21 10:15:00", "consumerConsentIp": "49.36.180.14" }</pre>
Response body	<pre>{ "success": "true", "leadId": "LN2026082100777" }</pre> "id" is accepted in place of "leadId". A refusal ("Not eligible", "'email' parameter is mandatory.") arrives with no lead id; the app raises the API's own wording so the user is told what to fix.
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/bl-create-lead' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{ ...the business lead payload above... }'</pre>

Update business lead

Endpoint	POST /v2/partner-auth/bl-update-lead/{lead_id}																														
Full URL	https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/bl-update-lead/{lead_id}																														
Called from	Business Loan lead form, when a business lead already exists — OffersRemoteDataSource.updateBusinessLead()																														
Request headers	Authorization: Bearer <token> Content-Type: application/json																														
Encryption	Request body is encrypted. The app sends {"data": "<envelope>"} (AES-256-GCM body, key wrapped with RSA-OAEP). The JSON shown on this page is what the app builds before encrypting it, and what the server reads after decrypting it. A reply that comes back as {"data": "<envelope>"} is decrypted by the app; the response shown is the decrypted JSON.																														
Path parameters	lead_id required The business lead id returned by bl-create-lead.																														
Request body	A strict subset of bl-create-lead: mobile, name, PAN, date of birth and the consent pair are fixed when the lead is created and this route does not accept them. <table><tr><td>email</td><td>required</td><td>string</td></tr><tr><td>pincode</td><td>required</td><td>string</td></tr><tr><td>monthlyIncome</td><td>required</td><td>number</td></tr><tr><td>businessRegistrationType</td><td>required</td><td>number</td></tr><tr><td>residenceType</td><td>required</td><td>number</td></tr><tr><td>businessEntityType</td><td>required</td><td>number</td></tr><tr><td>businessCurrentTurnover</td><td>required</td><td>number</td></tr><tr><td>businessYears</td><td>required</td><td>number</td></tr><tr><td>businessAccount</td><td>required</td><td>number</td></tr><tr><td>glIntent</td><td>optional</td><td>number</td></tr></table>	email	required	string	pincode	required	string	monthlyIncome	required	number	businessRegistrationType	required	number	residenceType	required	number	businessEntityType	required	number	businessCurrentTurnover	required	number	businessYears	required	number	businessAccount	required	number	glIntent	optional	number
email	required	string																													
pincode	required	string																													
monthlyIncome	required	number																													
businessRegistrationType	required	number																													
residenceType	required	number																													
businessEntityType	required	number																													
businessCurrentTurnover	required	number																													
businessYears	required	number																													
businessAccount	required	number																													
glIntent	optional	number																													

	<pre>{ "email": "rahul.sharma@example.com", "pincode": "400072", "monthlyIncome": 85000, "businessRegistrationType": 1, "residenceType": 2, "businessEntityType": 3, "businessCurrentTurnover": 3, "businessYears": 3, "businessAccount": 1 }</pre>
Response body	Not read by the app.
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/bl-update-lead/LN2026082100777' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{ ...the update payload above... }'</pre>

Get business offers

Endpoint	GET /v2/partner-auth/bl-get-offers/{lead_id}
Full URL	https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/bl-get-offers/{lead_id}
Called from	Business Loan offer carousel and the re-apply flow — OffersRemoteDataSource.getBusinessOffers()
Request headers	Authorization: Bearer <token> Content-Type: application/json
Encryption	No request body. The server always answers this route encrypted, as {"data": "<envelope>"}; the app decrypts it. The response shown on this page is the decrypted JSON.
Path parameters	lead_id required The business lead id.
Response body	The same structure as GET /partner-auth/get-offers/{lead_id} — success, leadId, offers[], unmatchedOffers[] — with every field alias described there. This route returns personal loan offers alongside the business ones; the app keeps only the business offers and drops the rest.
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/v2/partner-auth/bl-get-offers/LN2026082100777' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json'</pre>

7. Loan status and lender hand-off

Base URL: <https://uat.banxx.com:8000/in/home/app/api>

Offer summary

Endpoint	GET /partner-auth/get-summary/{creditprospect_uuid}
Full URL	https://uat.banxx.com:8000/in/home/app/api/partner-auth/get-summary/{creditprospect_uuid}
Called from	Loan status tracker — LoanApplicationService.fetchSummary()
Request headers	Authorization: Bearer <token> X-CSRF-TOKEN: (sent as an empty value)
Encryption	No request body. The server always answers this route encrypted, as {"data": "<envelope>"}; the app decrypts it. The response shown on this page is the decrypted JSON.
Path parameters	creditprospect_uuid required From the /user-mobile record.
Response body	<pre>{ "summary": { "offersTotal": 3, "maxLoanAmount": "500000", "minMPR": "1.04", "maxMPR": "1.75" }, "redirectionUrl": "https://uat.banxx.com/in/home/app/offers/cp_8f21d0b4e7a3" }</pre>
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/partner-auth/get-summary/cp_8f21d0b4e7a3' \ -H 'Authorization: Bearer <token>' \ -H 'X-CSRF-TOKEN:'</pre>

Gold loan hand-off

Endpoint	POST /gold-loans/{creditapp_uuid}
Full URL	https://uat.banxx.com:8000/in/home/app/api/gold-loans/{creditapp_uuid}
Called from	Loan status tracker and the gold offer card's Continue button — LoanApplicationService.submitGoldLoan()
Request headers	Authorization: Bearer <token> X-CSRF-TOKEN: (sent as an empty value)
Path parameters	creditapp_uuid required creditapp_uuid from GET /user-mobile/{credit_uid}. Only that record hands it out, so /user-mobile always runs first.
Request body	None.
Response body	<pre>{ "status": "true", "applicationId": 690289, "lenderId": 12, "redirect": "lenders-transaction/muthootgold?applicationId=690289" }</pre> Reads as a write but is idempotent — repeated calls return the same applicationId — so it is safe on every status refresh. `status` is "true" even when nothing was created, so applicationId > 0 is the only usable success signal. `redirect` is a path relative to https://uat.banxx.com/in/home/app (the same host on the default port; these paths 404 on :8000). The lender slug is read out of that path — muthootgold above — rather than from a lenderId map.
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/gold-loans/ca_5b7290ffe14c' \ -H 'Authorization: Bearer <token>' \ -H 'X-CSRF-TOKEN:'</pre>

Housing loan hand-off

Endpoint	POST /housing-loan/{creditapp_uuid}
Full URL	https://uat.banxx.com:8000/in/home/app/api/housing-loan/{creditapp_uuid}

Called from	Loan status tracker and the housing offer card's Continue button — LoanApplicationService.submitHousingLoan()
Request headers	Authorization: Bearer <token> X-CSRF-TOKEN: (sent as an empty value)
Path parameters	creditapp_uuid required creditapp_uuid from GET /user-mobile/{credit_uid}.
Request body	None.
Response body	Identical in shape to the gold hand-off — status, applicationId, lenderId, redirect — and read the same way.
cURL	curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/housing-loan/ca_5b7290ffe14c' \ -H 'Authorization: Bearer <token>' \ -H 'X-CSRF-TOKEN;'

8. Notifications

Base URL: `https://uat.banxx.com:8000/in/home/app/api`

Notification list

Endpoint	GET /banxx/notifications
Full URL	<code>https://uat.banxx.com:8000/in/home/app/api/banxx/notifications</code>
Called from	Notification dropdown and the notifications screen, and once per navbar mount for the unread badge — <code>NotificationsService.fetchPage()</code> / <code>refreshUnreadCount()</code>
Request headers	Authorization: Bearer <token> Accept: application/json X-CSRF-TOKEN: (sent as an empty value)
Query parameters	<code>page</code> optional 1-based page number. Default 1. <code>per_page</code> optional Rows per page. Default 20; clamped by the app to 1..100 because the server caps at 100 and clamps silently. <code>opened</code> optional true / false to filter by read state. Omitted for the mixed list the dropdown shows; sent as false for the badge-only call.
Response body	<pre>{ "data": [{ "notificationId": "ntf_00194823", "category": "loan_alerts", "title": "Gold loan application received", "text": "Your gold loan application has been sent...", "timestamp": "2026-08-20T09:15:40Z", "opened": false }], "pagination": { "current_page": 1, "last_page": 4, "has_more": true }, "total_unopened": 7 }</pre> <code>total_unopened</code> is account-wide, not per page — it is what the bell badge shows, which is why the badge refresh asks for <code>per_page=1</code>. The count is cached for 60 seconds. <code>`opened`</code> is accepted as a boolean, 0/1 or "true"/"1". <code>has_more</code> is trusted when sent; otherwise <code>current_page < last_page</code> is used. <code>`text`</code> is truncated here and complete in the fetch response. HTTP 400 means the account has no inbox yet — the app reads that as an empty list with a zero badge, not as an error. <code>`category`</code> is free text from the back office. The app picks a notification's icon by matching words in category and title (payment / upi / transfer / refund; bank / linked / consent; alert / card / loan / bill / due / emi / limit) and falls back to a Banxx badge, so a new category never breaks the list.
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/banxx/notifications? page=1&per_page=20' \ -H 'Authorization: Bearer <token>' \ -H 'Accept: application/json' \ -H 'X-CSRF-TOKEN;' # unread badge only curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/banxx/notifications? page=1&per_page=1&opened=false' \ -H 'Authorization: Bearer <token>' \ -H 'Accept: application/json'</pre>

Fetch one notification

Endpoint	GET /banxx/notifications/fetch/{notification_id}
Full URL	<code>https://uat.banxx.com:8000/in/home/app/api/banxx/notifications/fetch/{notification_id}</code>
Called from	Notification detail screen — <code>NotificationsService.openNotification()</code>

Request headers	Authorization: Bearer <token> Accept: application/json X-CSRF-TOKEN: (sent as an empty value)
Path parameters	notification_id required notificationId from the list response.
Response body	<pre>{ "data": { "notificationId": "ntf_00194823", "category": "loan_alerts", "title": "Gold loan application received", "text": "Your gold loan application has been sent to Muthoot Finance and is currently under review.", "timestamp": "2026-08-20T09:15:40Z", "opened": true } }</pre> Calling this marks the notification opened server-side — this request is what "reading" a notification means. The app drops its unread badge by one at the same moment rather than waiting for the next list fetch. A failure here costs the full body, not the screen: the detail view is already showing the list's copy of the title and text.
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/banxx/notifications/fetch/ ntf_00194823' \ -H 'Authorization: Bearer <token>' \ -H 'Accept: application/json' \ -H 'X-CSRF-TOKEN;'</pre>

9. Promotional banners

Base URL: <https://uat.banxx.com:8000/in/home/app/api>

Each banner can carry link, order and widget. The list is ordered by order first and by timestamp (newest first) only as a tiebreak.

Banner list

Endpoint	GET /banxx/banners
Full URL	https://uat.banxx.com:8000/in/home/app/api/banxx/banners
Called from	Dashboard only, through BannersService.refresh(), at three points: on mount — dashboard_screen.dart initState on foreground — returning to the app; cheap, the TTL usually serves it connection restored — refresh(force: true); the mount fetch is the one the dead connection ate, so cache is not worth serving Cached for 10 minutes in between. One request fills all four placements, and this is the app's only banner call on the dashboard path.
Request headers	Authorization: Bearer <token> Accept: application/json X-CSRF-TOKEN: (sent as an empty value)
Query parameters	active sent as 'true' on every call — only live banners are wanted.
Response body	<pre>{ "success": true, "data": [{ "bannerId": "5a11ple0-0000-4000-8000-000000000001", "image_url": "https://<bucket>/banner/1787827554.png", "name": "Popup 330x200", "title": "Popup banner", "text": "Opens over the dashboard on load, once a day", "type": "330x200", "widget": "Personal Loan", "link": "https://uat.banxx.com/in/home/app/offers", "order": 1, "active": true, "timestamp": "2026-08-27T10:05:00+05:30" }], "count": 5 }</pre> success must be a real boolean true; anything else leaves the currently loaded banners in place rather than blanking them. Aliases: bannerId / banner_id / id, image_url / imageUrl, link / url, order / position, widget / widget_name. A record with no id is dropped. `active` absent means shown (the list is already filtered). Ordering: `order` ascending first — a banner with no order sorts after every banner that has one — then `timestamp` newest first as a tiebreak. That is what decides the banner a single-banner slot shows.
Placement	The app picks the placement from `type` alone — nothing else routes it. The server enum is 284x167 330x200 354x227 354x100 popup. 354x227 Dashboard top: full-width card under Charles. A second banner at this size is placed lower, below the opening widget. 330x200 Popup over the dashboard, once per banner per day. The alias "popup" lands here too. 284x167 Slides inside a widget's offer carousel, spread through it (two offers, a banner, two offers, ...). `widget` narrows which carousel. 354x100 Sticky strip above the bottom navigation bar; several page through with arrows. Matching is lenient about how the size is written — "354 X 227", "354x227" and "354x227" all resolve. A type this build has no slot for is parsed and logged but never drawn, and so is a banner with no artwork or one switched off. Banners are cleared on logout.
Live, not sample	Every build calls this endpoint. The bundled sample response

	(lib/features/banners/data/sample_banners.dart) is opt-in through --dart-define=BANNER_SAMPLE_DATA=true, defaults to false, and is ANDed with kDebugMode — so a release build talks to the real API even with the define set. A slot with no banner at its size renders nothing; that is "no banner published at that size", not a failure. While the sample is on, the load logs a "SAMPLE DATA" line every time and the artwork is stamped SAMPLE BANNER — worth checking first when a banner published in the panel does not appear.
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/banxx/banners?active=true' \ -H 'Authorization: Bearer <token>' \ -H 'Accept: application/json' \ -H 'X-CSRF-TOKEN;'</pre>

10. AI Chat and Voice Command

Base URL: <https://uat.banxx.com:8000/in/home/app/api/chat/>

Every endpoint in this section and in sections 11 and 13 authenticates on the bearer token alone. No apikey and no phone number is sent on any of them.

Start conversation

Endpoint	POST chat/start
Full URL	https://uat.banxx.com:8000/in/home/app/api/chat/start
Called from	AI chat popup and the voice command card, on open — AiChatPopup._initializeChat(), VoiceCommandCard._initializeChat(), SupportScreen._startNewThread()
Request headers	Authorization: Bearer <token> Content-Type: application/json
Request body	{} An empty JSON object. The phone / phone_number field this used to carry is gone — the server reads the customer from the token.
Response body	{ "conversation_id": "cnv_7d4a19b2" } conversation_id is held for the life of the popup and sent with every message. A 200 carrying no conversation_id is a dead end: the popup still opens and the questions still load, but every send afterwards answers "Error: Conversation not initialized." A thread is titled server-side from its first message, so the conversation list is only worth re-reading after a message lands. The dashboard chat calls this on every open and keeps no history; only the Support screen lists past threads.
cURL	curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/chat/start' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{}'

Send message

Endpoint	POST chat/message
Full URL	https://uat.banxx.com:8000/in/home/app/api/chat/message
Called from	AI chat popup, voice command card and the support chat tab. A voice command sends its transcript here automatically.
Request headers	Authorization: Bearer <token> Content-Type: application/json
Request body	conversation_id required From chat/start. message required Free text — typed or dictated. { "conversation_id": "cnv_7d4a19b2", "message": "How much can I borrow against my gold?" }
Response body	{ "message": "Based on the current rate, you could borrow up to **75%** of your gold's value..." } Rendered as markdown in the chat bubble — bold, italic, inline code, headings, ordered and bulleted lists, rules and fenced code. A missing `message` shows "No response". The request is cancellable — the Stop button drops it silently, with no error bubble, and the user's own message stays in the thread because it did reach the server.
cURL	curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/chat/message' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{"conversation_id": "cnv_7d4a19b2", "message": "..."}'

Prepared questions

Endpoint	GET chat/questions
Full URL	https://uat.banxx.com:8000/in/home/app/api/chat/questions
Called from	AI chat popup and the support chat tab, on open. Shown as tappable cards before the user types anything; tapping one sends it immediately.
Request headers	Authorization: Bearer <token> Content-Type: application/json Changed: the bearer token is now required. This endpoint used to answer without one because the list was the same for everyone, and it 401s without it today.
Response body	<pre>{ "questions": ["What is my credit score?", "Which loan offers do I qualify for?", "How do I improve my eligibility?"] }</pre> A response with no `questions` key puts an error string in the list rather than leaving it empty, so a failure here is visible in the popup.
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/chat/questions' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json'</pre>

11. AI Advice

Base URL: https://uat.banxx.com:8000/in/home/app/api/chat/

AI advice

Endpoint	POST chat/advice
Full URL	https://uat.banxx.com:8000/in/home/app/api/chat/advice
Called from	AI advice card on the dashboard and the advice popup — AiAdviceItem.fetchLatest()
Request headers	Authorization: Bearer <token> Content-Type: application/json
Request body	{} An empty JSON object. No phone number is sent.
Response body	[{ "short": "Move your EMI date", "long": "Your salary credits on the 1st but your EMI debits on the 28th. Shifting the EMI to the 3rd would avoid the shortfall you hit in June and July.", "category": "Loans", "saving": "1200" }] The first item is used. A single object at the root is also accepted, as is the list nested under any of advice, advice_list, advices, data, results or items. Field aliases: short / title / advice / message for the headline, long / description / detail / body for the text, category / type, saving / amount. An empty list renders "No advice available right now."; a thrown failure renders a Try again.
cURL	curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/chat/advice' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{}'

12. Deals and Offers (vouchers)

Base URL: `https://uat.banxx.com:8000/in/home/app/api`

Get voucher brands

Endpoint	POST /vouchagram/getbrands
Full URL	<code>https://uat.banxx.com:8000/in/home/app/api/vouchagram/getbrands</code>
Called from	Deals & Offers card on the dashboard — <code>OffersRemoteDataSource.getDealsAndOffers()</code>
Request headers	Authorization: Bearer <token> Accept: application/json X-CSRF-TOKEN: (sent as an empty value)
Request body	None.
Response body	<pre>{ "success": true, "data": "[{\\"BrandName\\":\\"Amazon\\", \\"Brandtype\\":\\"E-Gift Voucher\\", \\"Category\\":\\"Shopping\\", \\"BrandImage\\":\\"https://cdn.../amazon.png\\"}]"]</pre> <code>`data`</code> currently arrives as the brand list re-encoded as a JSON string. The app peels the envelope and decodes strings in a loop (up to four levels), so a bare array, or <code>`data`</code> holding the list itself, works too — a backend that stops double-encoding will not blank the card. Fields read per brand: BrandName (title), Brandtype (tag and fallback description), Category, BrandImage (artwork), Descriptions (fallback). A BrandImage carrying a width with no height comes back cropped by the CDN and is normalised by the app. Never throws: an empty or failed response renders the card's empty state, and no token at all returns an empty list without a request. A 401 is refreshed and retried once like every other Banxx call.
Replaces	This endpoint fronts BulkGV. The app's direct BulkGV integration — <code>bulkGvBaseUrl</code> , the <code>gettoken</code> handshake, its credentials and the AES key and IV that decrypted the response — has been removed entirely; the backend holds those secrets and there is nothing left in the app to mint, cache or decrypt. The app never talks to <code>send.bulkgv.net</code> .
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/vouchagram/getbrands' \ -H 'Authorization: Bearer <token>' \ -H 'Accept: application/json' \ -H 'X-CSRF-TOKEN;'</pre>

13. Support — Help and Chat

Base URL: `https://uat.banxx.com:8000/in/home/app/api/chat/`

FAQ

Endpoint	GET chat/faq
Full URL	<code>https://uat.banxx.com:8000/in/home/app/api/chat/faq</code>
Called from	Support screen, Help tab — <code>SupportScreen._fetchFaq()</code>
Request headers	Authorization: Bearer <token> Content-Type: application/json Changed: the bearer token is now required. Like chat/questions this used to answer without any credential, and 401s without one today.
Response body	<pre>{ "faq": [{ "title": "Loans", "subtitles": [{ "subtitle": "Gold loans", "questions": [{ "question": "How is my gold valued?", "answer": "The lender values your gold at the prevailing rate..." }] }] }] }</pre> Sometimes delivered as a JSON string rather than an object; the app detects and decodes that case. Categories with no questions are dropped, and a "Banxx FAQ -" or "FAQs" prefix/suffix on a title is trimmed for display. A failure puts a retry on the tab rather than an empty list.
cURL	<code>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/chat/faq' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json'</code>

FAQ feedback

Endpoint	POST chat/faq-feedback
Full URL	<code>https://uat.banxx.com:8000/in/home/app/api/chat/faq-feedback</code>
Called from	Support screen, the thumbs up / down on an FAQ article — <code>SupportScreen._submitFaqFeedback()</code>
Request headers	Authorization: Bearer <token> Content-Type: application/json
Request body	title required FAQ category title. subtitle required FAQ group subtitle. question required The article question. reaction required The reaction recorded. <pre>{ "title": "Loans", "subtitle": "Gold loans", "question": "How is my gold valued?", "reaction": "up" }</pre> No phone number is sent; the server reads the customer from the token.
Response body	<pre>{ "message": "Thanks for your feedback!" }</pre> <code>`message`</code> is shown to the user when present; otherwise the app shows "Thanks for your feedback!". A failure shows "Could not submit feedback. Please try again." One reaction per article per session.

cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/chat/faq-feedback' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json' \ -d '{"title": "Loans", "subtitle": "Gold loans", "question": "How is my gold valued?", "reaction": "up"}'</pre>
-------------	---

Conversation list

Endpoint	GET chat/conversations
Full URL	https://uat.banxx.com:8000/in/home/app/api/chat/conversations
Called from	Support screen, the chat history drawer — SupportScreen._fetchConversations()
Request headers	Authorization: Bearer <token> Content-Type: application/json Changed: the path was conversations/{phone}. The phone segment is gone — the server reads the account from the token.
Response body	<pre>{ "conversations": [{ "id": 4192, "thread_id": "cnv_7d4a19b2", "created_at": "2026-08-18T14:22:03Z", "first_message": "How much can I borrow against my gold?" }] }</pre> first_message titles the drawer row; a thread with none shows "New Conversation". Re-read after a message is sent, since the server titles a thread from its first message. An empty list shows "No previous conversations".
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/chat/conversations' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json'</pre>

Conversation history

Endpoint	GET chat/history/{conversation_id}
Full URL	https://uat.banxx.com:8000/in/home/app/api/chat/history/{conversation_id}
Called from	Support screen, on opening a past thread — SupportScreen._fetchChatHistory()
Request headers	Authorization: Bearer <token> Content-Type: application/json
Path parameters	conversation_id required thread_id from the conversation list.
Response body	<pre>{ "messages": [{ "id": 88301, "role": "user", "content": "How much can I borrow against my gold?", "created_at": "2026-08-18T14:22:03Z" }, { "id": 88302, "role": "assistant", "content": "Based on the current rate...", "created_at": "2026-08-18T14:22:07Z" }] }</pre> role is user or assistant and decides which side the bubble renders on. Assistant content is markdown, the same as chat/message.
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/chat/history/cnv_7d4a19b2' \ -H 'Authorization: Bearer <token>' \ -H 'Content-Type: application/json'</pre>

14. Settings

Base URL: <https://uat.banxx.com:8000/in/home/app/api>

Get notification preferences

Endpoint	GET /banxx/notification-preferences
Full URL	https://uat.banxx.com:8000/in/home/app/api/banxx/notification-preferences
Called from	Settings > Notifications, on open — BanxxAuthService.getNotificationPreferences(). Held in an in-memory cache afterwards, so leaving Settings and returning does not re-fetch.
Request headers	Authorization: Bearer <token> X-CSRF-TOKEN: (sent as an empty value)
Response body	<pre>{ "data": { "payments": true, "bills_due_reminders": true, "credit_card_alerts": false, "loan_alerts": true, "critical_alerts": true } }</pre> The same fields are also accepted at the root. Each is read as true only on a real boolean true — a 1 or "true" reads as off. All five are stored and sent back, but this build shows only two rows: loan_alerts as the one switch the user can move, and critical_alerts as an always-on row whose control is disabled. The other three are loaded and saved untouched.
cURL	<pre>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/banxx/notification-preferences' \ -H 'Authorization: Bearer <token>' \ -H 'X-CSRF-TOKEN;'</pre>

Save notification preferences

Endpoint	POST /banxx/notification-preferences
Full URL	https://uat.banxx.com:8000/in/home/app/api/banxx/notification-preferences
Called from	Settings > Notifications, on each toggle — BanxxAuthService.saveNotificationPreferences()
Request headers	Authorization: Bearer <token> X-CSRF-TOKEN: (sent as an empty value) Content-Type: application/json
Request body	<pre>payments required boolean bills_due_reminders required boolean credit_card_alerts required boolean loan_alerts required boolean critical_alerts required boolean { "payments": true, "bills_due_reminders": true, "credit_card_alerts": false, "loan_alerts": true, "critical_alerts": true }</pre> All five are always sent, not just the one that changed. critical_alerts is always sent as true — the UI cannot turn it off.
Response body	<pre>{ "success": true }</pre> A "status" key is accepted in place of "success". If the response carries neither, any 2xx is taken as saved.
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/banxx/notification-preferences' \ -H 'Authorization: Bearer <token>' \ -H 'X-CSRF-TOKEN;'</pre>

	<pre>-H 'Content-Type: application/json' \ -d '{"payments": true, "bills_due_reminders": true, "credit_card_alerts": false, "loan_alerts": true, "critical_alerts": true}'</pre>
--	--

Delete account

Endpoint	POST /banxx-app/delete-account
Full URL	https://uat.banxx.com:8000/in/home/app/api/banxx-app/delete-account
Called from	Settings > Delete your account, after the confirmation dialog — BanxxAuthService.deleteAccount()
Request headers	Authorization: Bearer <token> X-CSRF-TOKEN: (sent as an empty value)
Request body	None. The server deletes the account that the bearer token belongs to, so no phone number is sent.
Response body	<pre>{ "success": true, "message": "Account deleted" }</pre> A "status" key is accepted in place of "success"; accepted values are true and 1, and anything else is treated as a failure and shows "Unable to delete your account right now. Please try again." On success the app wipes secure storage and every cache — profile, user record, banners, notification badge — and returns the user to login.
cURL	<pre>curl -X POST 'https://uat.banxx.com:8000/in/home/app/api/banxx-app/delete-account' \ -H 'Authorization: Bearer <token>' \ -H 'X-CSRF-TOKEN;'</pre>

15. Legal and compliance pages

Base URL: `https://uat.banxx.com:8000/in/home/app/api`

Legal page

Endpoint	GET /legal/{slug}
Full URL	<code>https://uat.banxx.com:8000/in/home/app/api/legal/{slug}</code>
Called from	Terms & legal screens (legal list and legal page view) — LegalService. The six pages are fetched one request per slug, in parallel, and cached in memory and on the device so a policy still shows with no network. A page that 404s is skipped; the rest still load.
Request headers	None. These pages are public — readable before login and by a store reviewer who never signs in — so no Authorization header is sent.
Path parameters	slug required One of: terms-of-use privacy-policy refund-policy cookie-policy disclaimer grievance-redressal-policy
Response body	<pre>{ "success": true, "data": { "title": "Privacy Policy", "content": "<p>This policy explains how BANXX handles your data...</p>" } }</pre> The app is read-only on these pages: no acceptance or create call exists. A single object under `data`, a list, or a bare object are all accepted, and the slug asked for is used when the body does not echo one.
cURL	<code>curl -X GET 'https://uat.banxx.com:8000/in/home/app/api/legal/privacy-policy'</code>

16. Payload encryption — how to encrypt and decrypt

Use this to build or test an encrypted request, or to read an encrypted reply (for example from a proxy capture). The endpoint pages show the plain JSON.

16.1 Keys and algorithm

Algorithm	Hybrid encryption. The JSON is encrypted with a one-off AES-256-GCM key, and that AES key is wrapped with RSA-2048 (OAEP, SHA-1 with MGF1-SHA-1). Only the small AES key goes through RSA, so any body size works.
Public key	Used to ENCRYPT. It is built into the app and the server wraps its replies to the same key pair. The key is not printed in this document; it is shared separately in a locked zip.
Private key	Used to DECRYPT. Decrypting a reply (and reading a request on the server) needs the private key that pairs with the public key, the one issued by Asset Direct. It is not printed in this document; it is shared separately in the locked zip. The app takes it from the build define BANXX_RSA_PRIVATE_KEY.

16.2 Encrypt a request (public key)

Step 1	Take the request body as JSON, exactly as shown on the endpoint's page (for example {"mobile_phone_number": 9876543210}), and serialise it to a UTF-8 string.
Step 2	Generate a fresh random AES-256 key (32 bytes) and a fresh random IV (12 bytes). Never reuse either — every request gets its own.
Step 3	Encrypt the JSON string with AES-256-GCM using that key and IV. No additional authenticated data. You get the ciphertext and a 16-byte authentication tag.
Step 4	Wrap the 32-byte AES key with the RSA public key: RSA-OAEP, hash SHA-1, MGF1 with SHA-1. The result is 256 bytes.
Step 5	Join the pieces in this order: 4-byte big-endian length of the wrapped key, the wrapped key, the IV, the tag, the ciphertext. The tag goes BEFORE the ciphertext — many libraries append it after, so move it.
Step 6	Base64-encode the joined bytes (standard alphabet, with padding).
Step 7	Send the request with Content-Type: application/json and the body {"data": "<base64 from step 6>"}. A GET has no body, so nothing is encrypted on the way out — never send a body on a GET (the server answers 403).

16.3 Decrypt a reply (private key)

Step 1	Check the reply: it is encrypted when the body has a `data` field holding a long base64 string (over 380 characters). Other keys beside it, such as success, are normal. A `data` that is a plain object or a short string is not encrypted — use it as it is.
Step 2	Base64-decode `data`.
Step 3	Read the first 4 bytes as a big-endian number N. The next N bytes are the wrapped AES key; then 12 bytes of IV; then 16 bytes of tag; everything left is the ciphertext.
Step 4	Unwrap the AES key with the RSA PRIVATE key: RSA-OAEP, SHA-1, MGF1 with SHA-1.
Step 5	Decrypt the ciphertext with AES-256-GCM using the unwrapped key, the IV and the tag. If the tag check fails, the message was altered or the wrong key was used — reject it.
Step 6	Decode the plaintext bytes as UTF-8 and parse the JSON. That JSON is the response shown on the endpoint's page.

16.4 Envelope layout

Bytes 0-3	Length N of the wrapped key, unsigned 32-bit, big-endian (256 for an RSA-2048 key).
Next N bytes	AES key, wrapped with RSA-OAEP (SHA-1).
Next 12 bytes	IV (nonce) for AES-GCM.
Next 16 bytes	AES-GCM authentication tag.
Remaining bytes	Ciphertext (same length as the JSON text).

Whole envelope	Base64 of all of the above, in the `data` field. Smallest possible envelope: 4 + 256 + 12 + 16 bytes.
-----------------------	---