FreedmAI Complete API Documentation

Microservices | APIs | Requests | Responses | Logic | MySQL Queries

Version 2.0.0 | September 19, 2025

🏗️ Microservices Overview

Service	Port	Base URL	Database Tables	Responsibilities
API Gateway	3000	http://localhost:3000	None (Proxy Only)	Request routing, service discovery, health monitoring
User Service	3001	http://localhost:3001	users, identities, kyc_summary	User management, profiles, KYC verification
OTP Service	3007	http://localhost:3007	otp_attempts, sessions	OTP generation/verification, JWT tokens, authentication
Notification Service	3006	http://localhost:3006	audit_logs	SMS, email notifications, delivery tracking

API 1: Send OTP

POST **/api/auth/send-otp**

Microservice: OTP Service (Port 3007)

Gateway URL: http://localhost:3000/api/auth/send-otp

Request

```
{ "phone_number": "+919876543210", "purpose": "registration" }
```

Response

```
{ "success": true, "data": { "verification_id": "550e8400-e29b-41d4-a716-446655440000", "expires_at": "2025-09-19T10:35:00Z", "otp_length": 6, "retry_after": 60 }, "message": "OTP sent successfully", "timestamp": "2025-09-19T10:30:00Z" }
```

Logic Implementation

> 1. Validate phone number format using regex: /^\+91[6-9]\d{9}$/

2. Check rate limiting: Max 3 OTP requests per 5 minutes per phone

3. Generate 6-digit random OTP: Math.floor(100000 + Math.random() * 900000)

4. Create unique verification_id using UUID v4

5. Hash OTP using SHA-256 with salt for secure storage

6. Store OTP attempt in database with 5-minute expiry

7. Call Notification Service to send SMS

8. Return verification_id to client for OTP verification

MySQL Queries

```
-- Rate limiting check SELECT COUNT(*) as attempt_count FROM otp_attempts WHERE identifier = '+919876543210' AND created_at > DATE_SUB(NOW(), INTERVAL 5 MINUTE); -- Insert OTP attempt INSERT INTO otp_attempts ( identifier, otp_hash, verification_id, type, expires_at, created_at ) VALUES ( '+919876543210', 'sha256_hashed_otp_with_salt', '550e8400-e29b-41d4-a716-446655440000', 'mobile_verification', DATE_ADD(NOW(), INTERVAL 5 MINUTE), NOW() );
```

🔗 API 2: Verify OTP

POST **/api/auth/verify-otp**

Microservice: OTP Service (Port 3007)

Gateway URL: http://localhost:3000/api/auth/verify-otp

Request

```
{ "phone_number": "+919876543210", "otp": "123456", "verification_id": "550e8400-e29b-41d4-a716-446655440000" }
```

Response

```
{ "success": true, "data": { "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...", "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...", "user": { "id": 12345, "phone": "+919876543210", "status": "onboarding", "current_step": "mobile_otp" } }, "message": "OTP verified successfully", "timestamp": "2025-09-19T10:30:00Z" }
```

Logic Implementation

1. Validate OTP format: 6 digits only

2. Retrieve OTP record using verification_id

3. Check OTP expiry and verification status

4. Hash provided OTP with same salt

5. Compare hashes using constant-time comparison (crypto.timingSafeEqual)

6. Mark OTP as verified in database

7. Create or update user record

8. Generate JWT access token (15 min expiry) and refresh token (30 days)

9. Create session record with tokens

10. Return tokens and user data

MySQL Queries

```
-- Verify OTP attempt SELECT * FROM otp_attempts WHERE verification_id = '550e8400-e29b-41d4-a716-446655440000' AND identifier = '+919876543210' AND expires_at > NOW() AND is_verified = FALSE AND attempts_count < 5; -- Mark OTP as verified UPDATE otp_attempts SET is_verified = TRUE, attempts_count = attempts_count + 1 WHERE verification_id = '550e8400-e29b-41d4-a716-446655440000'; -- Create or update user INSERT INTO users ( phone, status, current_step, created_at ) VALUES ( '+919876543210', 'onboarding', 'mobile_otp', NOW() ) ON DUPLICATE KEY UPDATE updated_at = NOW(), current_step = 'mobile_otp'; -- Create session INSERT INTO sessions ( user_id, session_token, access_token_hash, expires_at, created_at ) VALUES ( 12345, 'jwt_refresh_token_string', 'sha256_access_token_hash', DATE_ADD(NOW(), INTERVAL 30 DAY), NOW() );
```

API 3: User Registration

POST **/api/users/register**

Microservice: User Service (Port 3001)

Gateway URL: http://localhost:3000/api/users/register

Request

```
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9... Content-Type: application/json Body: { "full_name": "John Doe", "email": "john@example.com", "dob": "1990-01-15", "pincode": "110001" }
```

Response

```
{ "success": true, "data": { "user": { "id": 12345, "phone": "+919876543210", "email": "john@example.com", "full_name": "John Doe", "dob": "1990-01-15", "pincode": "110001", "status": "active", "current_step": "profile_confirmation" } }, "message": "User registered successfully", "timestamp": "2025-09-19T10:30:00Z" }
```

Logic Implementation

1. Verify JWT access token and extract user_id
2. Validate email format using regex
3. Validate pincode format (6 digits)
4. Validate date of birth format
5. Update user profile with provided information
6. Create identity record for email
7. Initialize KYC summary record
8. Update user status to 'active' and step to 'profile_confirmation'
9. Return updated user profile

MySQL Queries

```
-- Update user profile UPDATE users SET full_name = 'John Doe', email = 'john@example.com',
dob = '1990-01-15', pincode = '110001', current_step = 'profile_confirmation', status =
'active', updated_at = NOW() WHERE id = 12345; -- Insert email identity INSERT INTO
identities ( user_id, identity_type, identity_value, verification_status, created_at ) VALUES
( 12345, 'email', 'john@example.com', 'pending', NOW() ) ON DUPLICATE KEY UPDATE
identity_value = VALUES(identity_value), updated_at = NOW(); -- Initialize KYC summary INSERT
INTO kyc_summary ( user_id, kyc_status, verification_level, created_at ) VALUES ( 12345,
'pending', 'basic', NOW() ) ON DUPLICATE KEY UPDATE updated_at = NOW();
```

🏷 API 4: Get User Profile

GET **/api/users/profile**

Microservice: User Service (Port 3001)

Gateway URL: http://localhost:3000/api/users/profile

Request

```
Headers: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Response

```
{ "success": true, "data": { "user": { "id": 12345, "phone": "+919876543210", "email":
```

```
"john@example.com", "full_name": "John Doe", "dob": "1990-01-15", "pincode": "110001",
"status": "active", "current_step": "profile_confirmation", "kyc_status": "pending",
"verification_level": "basic", "identities": [ { "type": "email", "value":
"john@example.com", "status": "pending" } ] } }, "message": "Profile retrieved successfully",
"timestamp": "2025-09-19T10:30:00Z" }
```


Logic Implementation

1. Verify JWT access token and extract user_id
2. Retrieve user profile from database
3. Join with KYC summary to get verification status
4. Retrieve user identities (email, documents)
5. Combine all data into comprehensive profile
6. Return complete user profile with KYC status

MySQL Queries

```
-- Get user profile with KYC status SELECT u.*, k.kyc_status, k.verification_level,
k.completed_at FROM users u LEFT JOIN kyc_summary k ON u.id = k.user_id WHERE u.id = 12345; -
- Get user identities SELECT identity_type, identity_value, verification_status, verified_at
FROM identities WHERE user_id = 12345;
```

API 5: Refresh Token

POST **/api/auth/refresh**

Microservice: OTP Service (Port 3007)

Gateway URL: http://localhost:3000/api/auth/refresh

Request


```
{ "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..." }
```


Response


```
{ "success": true, "data": { "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
"expires_in": 900 }, "message": "Token refreshed successfully", "timestamp": "2025-09-
19T10:30:00Z" }
```


Logic Implementation

1. Verify refresh token signature and expiry

2. Extract user_id and session_id from token

3. Verify session is active in database

4. Generate new access token with 15-minute expiry

5. Update session with new access token hash

6. Return new access token

MySQL Queries

```
-- Verify active session SELECT * FROM sessions WHERE user_id = 12345 AND session_token = 'jwt_refresh_token_string' AND is_active = TRUE AND expires_at > NOW(); -- Update session with new access token UPDATE sessions SET access_token_hash = 'new_sha256_access_token_hash', updated_at = NOW() WHERE user_id = 12345 AND session_token = 'jwt_refresh_token_string';
```

API 6: Send SMS Notification

POST **/api/notifications/sms**

Microservice: Notification Service (Port 3006)

Gateway URL: http://localhost:3000/api/notifications/sms

Request

```
{ "mobile_number": "+919876543210", "template_type": "otp_verification", "variables": { "otp": "123456", "expiry_minutes": 5 } }
```

Response

```
{ "success": true, "data": { "message_id": "sms_12345_67890", "status": "sent", "mobile_number": "+919876543210" }, "message": "SMS sent successfully", "timestamp": "2025-09-19T10:30:00Z" }
```

Logic Implementation

1. Validate mobile number format

2. Select SMS template based on template_type

3. Replace template variables with actual values

4. Call SMS gateway API (Twilio/AWS SNS)

5. Generate unique message_id for tracking

6. Log SMS delivery in audit_logs

7. Return message_id and delivery status

MySQL Queries

```
-- Log SMS delivery in audit logs INSERT INTO audit_logs ( user_id, action, details,
ip_address, created_at ) VALUES ( 12345, 'sms_sent', JSON_OBJECT( 'message_id',
'sms_12345_67890', 'template_type', 'otp_verification', 'mobile_number', '+919876543210',
'status', 'sent' ), '192.168.1.100', NOW() );
```

API 7: Health Check

GET **/health**

Microservice: API Gateway (Port 3000)

Gateway URL: http://localhost:3000/health

Request

```
GET /health
```

Response

```
{ "status": "UP", "service": "FreedmAI API Gateway", "timestamp": "2025-09-19T10:30:00Z",
"version": "1.0.0", "services": { "user_service": { "status": "UP", "url":
"http://localhost:3001" }, "otp_service": { "status": "UP", "url": "http://localhost:3007" },
"notification_service": { "status": "UP", "url": "http://localhost:3006" } } }
```

Logic Implementation

1. Check API Gateway service status

2. Ping each microservice health endpoint

3. Collect response status from all services

4. Determine overall system health

5. Return aggregated health status

MySQL Queries

```
-- No direct database queries for health check -- Each service checks its own database connection: -- User Service database check SELECT 1 as health_check; -- OTP Service database check SELECT COUNT(*) as active_sessions FROM sessions WHERE is_active = TRUE AND expires_at > NOW();
```

🗄 Database Tables Summary

Table	Primary Key	Key Columns	Purpose
users	id (BIGINT)	phone, email, status, current_step	User profiles and onboarding status
sessions	id (BIGINT)	user_id, session_token, expires_at	JWT session management
otp_attempts	id (BIGINT)	identifier, verification_id, otp_hash	OTP generation and verification
identities	id (BIGINT)	user_id, identity_type, identity_value	User identity documents
kyc_summary	id (BIGINT)	user_id, kyc_status, verification_level	KYC verification status
audit_logs	id (BIGINT)	user_id, action, details	System audit trail

🔒 Security Features

- **OTP Security:** SHA-256 hashing, 5-minute expiry, rate limiting (3 per 5 min)
- **JWT Tokens:** 15-minute access tokens, 30-day refresh tokens
- **Session Management:** Secure session binding with token rotation
- **Input Validation:** Regex validation for phone, email, pincode
- **Rate Limiting:** Prevents OTP spam and brute force attacks
- **Audit Logging:** Complete activity tracking for security monitoring