Appendix C: Mock API Contract for Authentication Module

C.1 Overview

This appendix defines the REST API contract used for the case study authentication module. The contract specifies endpoints, request/response schemas, and expected behaviors that generated client-side code must correctly implement.

Base URL	https://api.example.com/v1
Content Type	application/json (all requests and responses)
Authentication	Bearer token in Authorization header for protected endpoints

C.2 Endpoints

C.2.1 User Registration

Endpoint: `POST /api/register`

Purpose: Create a new user account

Request Body:

```
{
  "email": "string (required, valid email format)",
  "password": "string (required, min 8 chars, uppercase, lowercase, number)",
  "confirmPassword": "string (required, must match password)"
}
```

Success Response:

Status	Body
201 Created	{ "success": true, "message": "User registered successfully", "userId": "string" }

Error Responses:

Status	Condition	Body
400 Bad Request	Validation failure	{ "success": false, "error": "VALIDATION_ERROR", "message": "string", "fields": {...} }
409 Conflict	Email already exists	{ "success": false, "error": "EMAIL_EXISTS", "message": "An account with this email already exists" }
500 Internal Server Error	Server failure	{ "success": false, "error": "SERVER_ERROR", "message": "An unexpected error occurred" }

C.2.2 User Login

Endpoint: `POST /api/login`

Purpose: Authenticate user and receive access token

Request Body:

```
{
  "email": "string (required)",
  "password": "string (required)"
```

```
}
```

Success Response:

Status	Body
200 OK	{ "success": true, "accessToken": "JWT", "refreshToken": "string", "expiresIn": 3600, "tokenType": "Bearer" }

Error Responses:

Status	Condition	Body
400 Bad Request	Missing fields	{ "success": false, "error": "VALIDATION_ERROR", "message": "Email and password are required" }
401 Unauthorized	Invalid credentials	{ "success": false, "error": "INVALID_CREDENTIALS", "message": "Invalid email or password" }
429 Too Many Requests	Rate limit exceeded	{ "success": false, "error": "RATE_LIMITED", "message": "Too many login attempts", "retryAfter": 300 }

C.2.3 Token Refresh

Endpoint: `POST /api/refresh`

Purpose: Obtain new access token using refresh token

Request Body:

```
{ "refreshToken": "string (required)" }
```

Success Response:

Status	Body
200 OK	{ "success": true, "accessToken": "JWT", "expiresIn": 3600, "tokenType": "Bearer" }

Error Responses:

Status	Condition	Body
400 Bad Request	Missing refresh token	{ "success": false, "error": "VALIDATION_ERROR", "message": "Refresh token is required" }
401 Unauthorized	Invalid/expired token	{ "success": false, "error": "INVALID_TOKEN", "message": "Refresh token is invalid or expired" }

C.2.4 Logout

Endpoint: `POST /api/logout`

Purpose: Invalidate current session and tokens

Headers:

```
Authorization: Bearer <accessToken>
```

Request Body:

```
{ "refreshToken": "string (optional)" }
```

Success Response:

Status	Body
200 OK	{ "success": true, "message": "Logged out successfully" }

C.2.5 Protected Resource (Example)

Endpoint: `GET /api/protected/profile`

Purpose: Retrieve authenticated user's profile (demonstrates protected route access)

Headers:

```
Authorization: Bearer <accessToken>
```

Success Response:

Status	Body
200 OK	{ "success": true, "data": { "userId": "string", "email": "string", "createdAt": "ISO 8601" } }

Error Responses:

Status	Condition	Body
401 Unauthorized	Missing token	{ "success": false, "error": "UNAUTHORIZED", "message": "Access token required" }
401 Unauthorized	Expired token	{ "success": false, "error": "TOKEN_EXPIRED", "message": "Access token has expired" }
403 Forbidden	Invalid token	{ "success": false, "error": "FORBIDDEN", "message": "Access token is invalid" }

C.3 Token Specification

Access Token Format: JSON Web Token (JWT)

JWT Structure:

```
{
  "header": { "alg": "HS256", "typ": "JWT" },
  "payload": { "sub": "userId", "email": "user@example.com", "iat": 1234567890, "exp": 1234571490 }
}
```

Token Lifetimes:

Token Type	Lifetime
Access Token	1 hour (3600 seconds)
Refresh Token	7 days (604800 seconds)

Storage Recommendations:

Storage Option	Security Consideration
localStorage	Vulnerable to XSS; not recommended for sensitive tokens
sessionStorage	Cleared on tab close; vulnerable to XSS
httpOnly cookie	Preferred for access tokens; not accessible via JavaScript
Memory (variable)	Most secure for short-lived tokens; lost on page refresh

C.4 Validation Rules

Email Validation:

• Must be valid email format (RFC 5322 compliant)
• Maximum length: 254 characters

Password Validation:

• Minimum length: 8 characters
• Maximum length: 128 characters

• Must contain at least one uppercase letter (A-Z)
• Must contain at least one lowercase letter (a-z)
• Must contain at least one digit (0-9)
• Special characters permitted but not required

C.5 Error Code Reference

Error Code	HTTP Status	Description
VALIDATION_ERROR	400	Request body failed validation
EMAIL_EXISTS	409	Registration attempted with existing email
INVALID_CREDENTIALS	401	Login failed due to wrong email or password
UNAUTHORIZED	401	Request missing valid authentication
TOKEN_EXPIRED	401	Access token has expired
INVALID_TOKEN	401	Token is malformed or invalid
FORBIDDEN	403	Token valid but insufficient permissions
RATE_LIMITED	429	Too many requests from client
SERVER_ERROR	500	Unexpected server-side failure

C.6 Mock Server Implementation Note

For case study execution, a mock server implementing this contract can be created using:

• JSON Server with custom middleware
• MSW (Mock Service Worker) for browser-based mocking
• Express.js minimal implementation

The mock server must:

1. Validate request bodies against specified schemas
2. Return appropriate success/error responses
3. Generate valid JWT tokens (can use static test tokens for deterministic testing)
4. Enforce token validation on protected endpoints

Summary

Component	Count
Endpoints	5
Error codes	9
Token types	2 (access, refresh)
Validation rules	2 (email, password)