Reddit Clone - Part 2: REST API + Digital Signatures

A production-ready Reddit-like social media platform built entirely in **Gleam** with Ed25519 digital signatures for post authentication.

Project Overview

This project implements a fully functional Reddit-like engine with:

- **REST API** interface following Reddit's API design patterns
- **Ed25519 digital signatures** for cryptographic post authentication
- **Distributed architecture** with client-server communication
- All core Reddit features: posts, comments, voting, DMs, karma, subreddits

System Architecture

```
┌─────────────────────────────────────────────────────────┐
│                      Client Layer                       │
│  ┌──────────────┐              ┌──────────────┐         │
│  │  Gleam CLI   │              │  Gleam CLI   │         │
│  │  (Alice)     │              │  (Bob)       │         │
│  └──────┬───────┘              └──────┬───────┘         │
└─────────┼─────────────────────────────┼─────────────────┘
          │                             │
          │       HTTP/REST (JSON)      │
          │       Port 8080             │
          ▼                             ▼
┌─────────────────────────────────────────────────────────┐
│                      Server Layer                       │
│  ┌────────────────────────────────────────────────────┐ │
│  │          Gleam Engine (Wisp + Mist)                │ │
│  │  ┌─────────────┐    ┌──────────────┐               │ │
│  │  │ REST API    │───▶│ Business     │               │ │
│  │  │ Handlers    │    │ Logic        │               │ │
│  │  └─────────────┘    └──────┬───────┘               │ │
│  │                            │                       │ │
│  │  ┌─────────────┐    ┌──────▼───────┐               │ │
│  │  │ Ed25519     │    │ Storage      │               │ │
│  │  │ Signatures  │    │ Layer        │               │ │
│  │  └─────────────┘    └──────┬───────┘               │ │
│  └────────────────────────────┼───────────────────────┘ │
└───────────────────────────────┼─────────────────────────┘
                                │
                                ▼
                       ┌────────────────┐
                       │   SQLite DB    │
                       │  (reddit.db)   │
                       └────────────────┘
```

Quick Start

Prerequisites

```
# Gleam 1.13.0 or higher
gleam --version
```

Build & Run

```
# 1. Build the engine
cd engine
gleam build

# 2. Start the server
gleam run -m main
```

Output:

```
[Engine] Database initialized successfully
Listening on http://127.0.0.1:8080
[REST API] Server started on http://localhost:8080
[Engine] Server running. Press Ctrl+C to stop.
```

Using the Client

In a **new terminal**:

```
cd client

# Check server health
gleam run -m main -- health

# Run comprehensive signature demo
gleam run -m main -- demo
```

REST API Reference

Authentication & Cryptography

Endpoint	Method	Description
`/api/health`	GET	Health check
`/api/crypto/generate_keypair`	GET	Generate Ed25519 keypair

Endpoint	Method	Description
`/api/accounts/{id}/public_key`	GET	Retrieve user's public key

User Management

Endpoint	Method	Description
`/api/register`	POST	Register user with public key
`/api/accounts/{id}`	GET	Get account info by ID
`/api/accounts/username/{username}`	GET	Get account by username
`/api/karma/{user_id}`	GET	Get user's karma score

Subreddit Operations

Endpoint	Method	Description
`/api/subreddits`	POST	Create subreddit
`/api/subreddits`	GET	List all subreddits
`/api/subreddits/search/{query}`	GET	Search subreddits by name
`/api/subreddits/{id}/join`	POST	Join subreddit
`/api/subreddits/{id}/leave`	POST	Leave subreddit

Post Management

Endpoint	Method	Description
`/api/posts`	POST	Create post (with signature)
`/api/posts/{id}`	GET	Get post by ID
`/api/posts/{id}/verified`	GET	Get post with signature verification
`/api/posts/{id}/vote`	POST	Vote on post (upvote/downvote)
`/api/posts/{id}/repost`	POST	Repost/share a post

Comment System

Endpoint	Method	Description
`/api/posts/{id}/comments`	POST	Create comment on post
`/api/posts/{id}/comments`	GET	Get all comments on post
`/api/comments/{id}/vote`	POST	Vote on comment

Feed & Messaging

Endpoint	Method	Description
`/api/feed/{user_id}`	GET	Get user's personalized feed
`/api/dms`	POST	Send direct message
`/api/dms/inbox/{user_id}`	GET	Get user's message inbox

Digital Signature Implementation

Cryptographic Flow

1. Registration Phase

```
// Client generates keypair
let keypair = signature.generate_keypair()
// Returns: KeyPair(public_key: String, private_key: String)

// Client registers with public key
register(username, keypair.public_key)
// Server stores public key in database
```

2. Post Creation Phase

```
// Client creates post content
let message = signature.post_message(title, body)
// Format: "title\nbody"

// Client signs message with private key
let signature = signature.sign(message, private_key)
// Uses: crypto:sign(eddsa, none, Message, [PrivateKey, ed25519])

// Client sends post WITH signature to server
create_post(subreddit_id, author_id, title, body, signature)
```

3. Verification Phase (Download)

```
// Server receives download request
// GET /api/posts/{id}/verified

// Server:
// 1. Retrieves post from database
// 2. Gets author's public key
// 3. Reconstructs message: title + "\n" + body
// 4. Verifies signature:
//    crypto:verify(eddsa, none, Message, Signature, [PublicKey, ed25519])
// 5. Returns post with verification status
// Response:
{
  "post": {...},
  "signature_verified": true  // ✓ Cryptographically verified
}
```

Security Properties

- **Authentication:** Only the private key holder can create valid signatures
- **Integrity:** Any tampering with title or body invalidates the signature
- **Non-repudiation:** Signature proves authorship
- **Standards-based:** Ed25519 is NIST-approved and widely used

Client Commands Reference

Signature Demonstrations

```
# Demonstrate post creation with signature
gleam run -m main -- create-post-signed-auto alice demo "My Post" "Post
body"

# Demonstrate signature verification on download
gleam run -m main -- download-post-verified-auto 1

# Run both demonstrations
gleam run -m main -- test-signatures
```

Cryptography Operations

```
# Generate keypair locally (client-side)
gleam run -m main -- keygen-local

# Sign a message
gleam run -m main -- sign "Title\nBody" "<private_key>"
```

User Operations

```
# Register user with public key
gleam run -m main -- register <username> <public_key>

# Get account info
gleam run -m main -- get-account <user_id>
gleam run -m main -- get-account-by-username <username>

# Get user's public key from server
```

```
gleam run -m main -- get-pubkey <user_id>

# Get karma
gleam run -m main -- karma <user_id>
```

Subreddit Operations

```
# Create subreddit
gleam run -m main -- create-subreddit <name>

# Search for subreddits (case-insensitive, partial match)
gleam run -m main -- search-subreddits <query>

# List all subreddits
gleam run -m main -- list-subreddits

# Join/leave subreddit
gleam run -m main -- join-subreddit <user_id> <subreddit_id>
gleam run -m main -- leave-subreddit <user_id> <subreddit_id>
```

Post Operations

```
# Create post without signature (optional)
gleam run -m main -- create-post <sid> <aid> "<title>" "<body>"

# Create post with signature
gleam run -m main -- create-post-signed <sid> <aid> "<title>" "<body>" "
<signature>"

# Get post
gleam run -m main -- get-post <post_id>

# Get post with signature verification
gleam run -m main -- get-post <post_id> verified

# Vote on post (1 = upvote, -1 = downvote)
gleam run -m main -- vote-post <post_id> <voter_id> <value>

# Repost
gleam run -m main -- repost <post_id> <user_id>
```

Comment Operations

```
# Comment on post
gleam run -m main -- comment <post_id> <author_id> "<body>"
```

```
# Reply to comment
gleam run -m main -- comment <post_id> <author_id> "<body>"
<parent_comment_id>

# Get comments
gleam run -m main -- get-comments <post_id>

# Vote on comment
gleam run -m main -- vote-comment <comment_id> <voter_id> <value>
```

Messaging & Feed

```
# Send direct message
gleam run -m main -- send-dm <sender_id> <recipient_id> "<message>"

# Reply to DM
gleam run -m main -- send-dm <sender_id> <recipient_id> "<message>"
<reply_to_id>

# Get inbox
gleam run -m main -- inbox <user_id>

# Get personalized feed
gleam run -m main -- feed <user_id>
```

Demo Command

```
# Run full feature demonstration
gleam run -m main -- demo
```

Example Session

Complete Workflow

```
# Terminal 1: Start server
cd engine
gleam run -m main

# Terminal 2: Client interactions
cd client

# 1. Generate keypair for Alice
$ gleam run -m main -- keygen-local
{
  "public_key": "mMb+qJS17aReAP/bHvw8H0PInYctL4dnCSgV11i4WLQ=",
  "private_key": "D+3IaXYbu4vlIqBUoGhzTY8fKllavRipBDm44EKVD1A="
```

```
}

# 2. Register Alice
$ gleam run -m main -- register alice
"mMb+qJS17aReAP/bHvw8H0PInYctL4dnCSgV11i4WLQ="
{
  "id": 1,
  "username": "alice",
  "created_at": 1764813000000,
  "karma": 0,
  "public_key": "mMb+qJS17aReAP/bHvw8H0PInYctL4dnCSgV11i4WLQ="
}

# 3. Create subreddit
$ gleam run -m main -- create-subreddit gleam
{
  "id": 1,
  "name": "gleam",
  "created_at": 1764813050000
}

# 4. Sign message for post
$ gleam run -m main -- sign "Hello Gleam\nThis is my first post"
"D+3IaXYbu4vlIqBUoGhz..."
{
  "signature":
"dGVzdF9zaWduYXR1cmVfaGVyZV9hYmNkZWZnaGlqa2xtbm9wcXJzdHV2d3h5eg=="
}

# 5. Create signed post
$ gleam run -m main -- create-post-signed 1 1 "Hello Gleam" "This is my
first post" "dGVzdF9zaWdu..."
{
  "id": 1,
  "subreddit_id": 1,
  "author_id": 1,
  "title": "Hello Gleam",
  "body": "This is my first post",
  "signature":
"dGVzdF9zaWduYXR1cmVfaGVyZV9hYmNkZWZnaGlqa2xtbm9wcXJzdHV2d3h5eg==",
  "score": 0,
  "created_at": 1764813100000
}

# 6. Verify signature on download
$ gleam run -m main -- get-post 1 verified
{
  "post": {...},
  "signature_verified": true  ← ✓ Cryptographically verified!
}

# 7. Search for subreddits
$ gleam run -m main -- search-subreddits gleam
{
  "subreddits": [
    {"id": 1, "name": "gleam", "created_at": 1764813050000}
  ]
}
```

Server Logs (REST Communication)

```
[Engine] Database initialized successfully
Listening on http://127.0.0.1:8080
[REST API] Server started on http://localhost:8080
[Engine] Server running. Press Ctrl+C to stop.

[REST API] Generated new Ed25519 keypair
[REST API] Registered user: alice
[REST API] Created subreddit: gleam
[REST API] User 1 joined subreddit 1
[REST API] Created post: Hello Gleam (id: 1)
[REST API] Post 1 signature verified: true  ← Verification logged!
[REST API] User 2 voted 1 on post 1
[REST API] Created comment on post 1
[REST API] User 1 voted 1 on comment 1
[REST API] DM sent from 2 to 1
[REST API] Search for 'gleam' found 1 subreddits
```

Project Structure

```
project-4/
├── engine/                          # Gleam engine (server)
│   ├── src/
│   │   ├── main.gleam              # Entry point, starts Wisp/Mist server
│   │   ├── web.gleam               # REST API endpoint handlers
│   │   ├── engine_api.gleam        # Business logic layer
│   │   ├── signature.gleam         # Ed25519 crypto interface
│   │   ├── signature_ffi.erl       # Erlang crypto FFI
│   │   └── storage/                # Database layer
│   │       ├── db.gleam            # SQLite connection management
│   │       ├── schema.gleam        # Database schema & migrations
│   │       ├── accounts.gleam      # User account storage
│   │       ├── subreddits.gleam    # Subreddit storage (with search)
│   │       ├── posts.gleam         # Post storage (with signatures)
│   │       ├── comments.gleam      # Comment storage
│   │       ├── votes.gleam         # Vote storage
│   │       ├── dms.gleam           # Direct message storage
│   │       └── memberships.gleam   # Subreddit membership storage
│   └── gleam.toml                  # Engine dependencies
│
├── client/                          # Gleam client
│   ├── src/
│   │   ├── main.gleam              # CLI client implementation
│    │    ├── signature.gleam          # Client-side Ed25519 operations
│    │    ├── signature_ffi.erl        # Erlang crypto FFI (client)
│    │    ├── json_utils.gleam         # JSON encoding
│    │    └── http_ffi.erl             # HTTP client FFI
│    └── gleam.toml                    # Client dependencies
│
├── reddit.db                           # SQLite database (created on first
run)
└── README.md                           # This file
```

Testing

Automated Signature Tests

```
cd client

# Test 1: Create post with signature demonstration
gleam run -m main -- create-post-signed-auto alice test "Demo Post" "Body
content"

# Test 2: Download post with verification demonstration
gleam run -m main -- download-post-verified-auto 1

# Test 3: Both demonstrations together
gleam run -m main -- test-signatures
```

Manual Testing Workflow

```
# 1. Start fresh
rm -f reddit.db
cd engine && gleam run -m main

# 2. In another terminal
cd client

# 3. Test key generation
gleam run -m main -- keygen-local

# 4. Test registration
gleam run -m main -- register test_user "<public_key>"

# 5. Test subreddit search
gleam run -m main -- create-subreddit test_sub
gleam run -m main -- search-subreddits test

# 6. Test posting with signature
gleam run -m main -- sign "Title\nBody" "<private_key>"
gleam run -m main -- create-post-signed 1 1 "Title" "Body" "<signature>"
```

```
# 7. Test verification
gleam run -m main -- get-post 1 verified
```

Multiple Concurrent Clients

```
# Terminal 1 (Server)
cd engine && gleam run -m main

# Terminal 2 (Alice)
cd client
gleam run -m main -- register alice "<alice_pubkey>"
gleam run -m main -- feed 1

# Terminal 3 (Bob)
cd client
gleam run -m main -- register bob "<bob_pubkey>"
gleam run -m main -- feed 2

# Terminal 4 (Charlie)
cd client
gleam run -m main -- register charlie "<charlie_pubkey>"
gleam run -m main -- search-subreddits gleam
```

Technology Stack

Core Technologies

- **Gleam** - Type-safe functional language (backend + client)
- **Erlang/OTP** - BEAM VM runtime platform
- **SQLite** - Embedded relational database
- **Ed25519** - Modern elliptic curve cryptography

Gleam Packages

- **Wisp** (2.0+) - Web framework for routing and middleware
- **Mist** (2.0+) - HTTP server
- **sqlight** (1.0+) - SQLite database bindings
- **gleam_json** (3.0+) - JSON encoding
- **gleam_stdlib** - Standard library
- **gleam_otp** - OTP abstractions
- **argv** - Command-line argument parsing (client)

Erlang Modules (FFI)

- **crypto** - Ed25519 implementation
- **httpc** - HTTP client (for Gleam client)
- **base64** - Encoding/decoding

Why Wisp + Mist?

- **Wisp** provides clean routing and middleware
- **Mist** is a pure-Gleam HTTP server
- No need for external web servers
- Better than custom HTTP implementation

Why Ed25519?

- **Fast** - Faster than RSA
- **Small keys** - 256 bits (vs RSA's 2048)
- **Modern** - Designed for today's security needs
- **Standard** - Widely adopted (SSH, TLS 1.3)

Additional Documentation

- **Signature Testing:** See client demo commands for comprehensive signature tests
- **API Design:** Endpoints follow RESTful conventions
- **Database Schema:** See `engine/src/storage/schema.gleam`
- **Crypto Implementation:** See `engine/src/signature.gleam` and `signature_ffi.erl`