■■■■■■■■■■ ■■■■■■■■■■ ■■■■

■■■■■■■■■■ «PIXORA FACE»

■■■ ■■■■■■■■■■■ ■■■■■■■■■■■ ■ ■■■■■■■■■■

■■■■■■■■■: PIXORA FACE - ■■■■■■■■ ■■■■■■■ ■■■■■■■■■■■ ■■ ■■■■

■■■■■■■■■■■■■■■■: PIXORASOFT

■■■■ ■■■■■■■■■■: 19 ■■■■■■■■ 2026 ■.

1. ■■■■■■■ ■■■■■■■■■■■■■■■ ■■■■ ■ ■■■■■■■■■■■ ■■■■■■■■■■■■

```
    # ■■■■■■■■ ■■■■■■■■■■■ ■■■■■■■■■■■ ■■■■■■■■■■■■ ■■■■■ ■■ ■■■■■■■■■■■■■
    # ■■■■■■■■■■■■ ■■■■■■■ InsightFace buffalo_l ■■■■ ■■■■■■■■■■■ 512-■■■■■■■■ ■■■■■■■■
    def get_embeddings(self, image_data: bytes):
        """
        ■■■■■■■■■■■■ ■■■■■■■■■■■■■ ■■■■ ■■ ■■■■■■■■■■■■■
        ■■■■■■■■■ ■■■■■■■■■ ■■■■■■■■ PIXORA FACE.
        """
        if not self.initialized or self.app is None:
            raise RuntimeError("■■■■■■■ ■■■■■■■■■■■■■■■ ■■■■ ■■ ■■■■■■■■■■■■■■■■■")

        # ■■■■■■■■■■■■■ ■■■■■■■ ■ ■■■■■■■ ■■■■■■■■■■■■■
        img = self._bytes_to_image(image_data)

        # ■■■■■■■■■ ■■■■ ■ ■■■■■■■■■■■■ ■■■■■■■■■■■■■ ■ ■■■■■■■■ buffalo_l ■■■■■■■
        faces = self.app.get(img)

        if len(faces) == 0:
            return []

        # ■■■■■■■■■■■■ ■■■■■■■■■■■■■ ■ ■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■
        results = []
        for face in faces:
            embedding = face.embedding  # 512-■■■■■■■ ■■■■■■■
            confidence = float(face.det_score)  # ■■■■■■■■■■■■■ ■■■■■■■■■
            results.append((embedding, confidence))

        return results

    # ■■■■■■■■ ■■■■■■■■■■ ■■■■■ ■■■■■■■■■■ ■■■■■■■■■■■■■■ ■■■■
    def compare_embeddings(self, embedding1, embedding2):
        """■■■■■■■■■■ ■■■■■■■■■■■■■ ■ ■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■ ■■■■■■■■■"""
        # ■■■■■■■■■■■■■■ ■■■■■■■■■
        norm1 = np.linalg.norm(embedding1)
        norm2 = np.linalg.norm(embedding2)

        if norm1 == 0 or norm2 == 0:
            return 0.0

        # ■■■■■■■■■■■■ ■■■■■■■■■
        dot_product = np.dot(embedding1, embedding2)
        similarity = dot_product / (norm1 * norm2)

        # ■■■■■■■■■■■■■■■■■ ■■ [-1, 1] ■ [0, 1]
        normalized_similarity = (similarity + 1) / 2

        return float(normalized_similarity)
```

2. ■■■■■■■ ■■■■■■■■■■■■ ■■■■■■■■ ■ ■■■■■ ■■■■■■■■ POSTGRESQL

```
# ■■■■■■■■ ■■■■■■■ ■■■■■■■■ ■■■■ ■ ■■■■■■■■■■ ■■■■■ ■■■■■■■
# ■■■■■■■■■■■■ ■■■■■■■■■■■■ pgvector ■■■■ ■■■■■■■■■■■■■■ ■■■■■■■■ ■■ ■■■■■■■■■
async def search_faces_in_session(session_id, file, threshold=0.5, limit=50):
    """
    ■■■■■■ ■■■■■■■■ ■■■■ ■ ■■■■■■■ ■■■■■■■■■■■■ (■■■■■■■■■ ■■■■■■■■ FacePass)

    ■■■■■■■■■:
    1. ■■■■■■■■■■ ■■■■■■■■■■ ■■ ■■■■■■■■■■■■■■ ■■■■■■
    2. ■■■■■ ■■■■■■■■ ■■■■■ ■ ■■■■■■■■■■ ■■ ■ ■■■■■■■■■ ■■ ■■■■■■■
    3. ■■■■■■■■■■■■ ■■■■■■■■■■■■ ■■■■■ ■■■■■■■ ■■■■■■■■■
    """

    # ■■■■■■■■■■■■ ■■■■■■■■■■■■ ■■ ■■■■■■■■■■■■■■ ■■■■■■
    face_service = get_face_recognition_service()
    query_embedding, confidence = face_service.extract_single_embedding(file_data)

    if query_embedding is None:
        return {"matches": [], "message": "■■■■■ ■■ ■■■■■■■■■■■■"}

    # ■■■■■■■■■■■■■■ ■■■■■■■■■■■■ ■■■■ ■■■■■■■■■■■■■■■■■ ■■■■■■■■ ■■■■■■■■■
    embedding_norm = np.linalg.norm(query_embedding)
    if embedding_norm > 0:
        query_embedding = query_embedding / embedding_norm

    # ■■■■■■■■■■■■■ ■■■■■■■■■■■■ ■ ■■■■■■■■■■ ■■■■■■■ ■■■■ SQL ■■■■■■■■
    query_embedding_str = '[' + ','.join(map(str, query_embedding.tolist())) + ']'

    # ■■■■■■■■■■ ■■■■■■ ■■■■■■■■■ ■ ■■■■■■■■■■■■■■■■■ ■■■■■■■■■■ <=>
    # ■■■■■■■■: ■■■■■■■■■ = 1 - ■■■■■■■■■■■■
    query = text("""
        SELECT
            fe.photo_id,
            1 - (fe.embedding <=> :query_embedding) as similarity
        FROM face_embeddings fe
        WHERE fe.session_id = :session_id
            AND (1 - (fe.embedding <=> :query_embedding)) >= :threshold
        ORDER BY similarity DESC
        LIMIT :limit
    """)

    result = vector_db.execute(query, {
        "query_embedding": query_embedding_str,
        "session_id": session_id,
        "threshold": threshold,
        "limit": limit
    })

    # ■■■■■ ID ■■■■■■■■■■■■ ■ ■■■■■■■■■■■■■■■ ■■■■■■■■■
    photo_matches = []
    for row in result:
        photo_matches.append({
            "photo_id": row[0],
            "similarity": float(row[1])
        })

    return {"matches": photo_matches}
```

3. ███████ ████████████████ ████████████ ████████████

```
# ████████ ██████████ ██████ ████████████ ███ ████████████ ■ ██████████ ████████████
def process_single_photo(self, s3_key, session_id, vector_db):
    """
    ██████████ ██████ ████████████: █████████, ██████████ ████████████, ██████████

    █████████ PIXORA FACE ███ ████████████████ ████████████:
    1. ██████████ █████ ██ ██████████ ██████████
    2. ██████████ ███ ████ ■ ██ ██████████ ████████████
    3. ████████████ ████████ ███ ████████████████ ███████
    4. ██████████ ■ ██████████ ████ ███████ PostgreSQL
    """

    # ████████████ ID ████████████ ██ S3 ██████
    photo_id = self.extract_photo_id_from_s3_key(s3_key)
    if not photo_id:
        return False, f"██ ████████ ████████ ID ████"

    # █████████, ██ ████████████ ██ ███ ███ ████████████
    existing = vector_db.query(FaceEmbedding).filter(
        FaceEmbedding.photo_id == photo_id,
        FaceEmbedding.session_id == session_id
    ).first()

    if existing:
        return True, None  # ███ ████████████

    # █████████ ████████████ ██ S3
    photo_data = download_image(s3_key)

    # ████████████ ████████████ ███
    embeddings = self.face_service.get_embeddings(photo_data)

    if not embeddings:
        return True, None  # ████ ██ ████████ - ██ ███████

    # ████████████ ████████████ ■ ████ ███████
    for embedding_vector, confidence in embeddings:
        # ██████████████ ████████████
        embedding_norm = np.linalg.norm(embedding_vector)
        if embedding_norm > 0:
            normalized_embedding = embedding_vector / embedding_norm
        else:
            normalized_embedding = embedding_vector

        # █████████ ███████ ■ ██████████ ██
        face_embedding = FaceEmbedding(
            photo_id=photo_id,
            session_id=session_id,
            embedding=normalized_embedding.tolist(),
            confidence=confidence
        )

        vector_db.add(face_embedding)

    # █████████ ████████████
    vector_db.commit()

    return True, None
```

4. ■■■■■■■■■■■■■■ ■■■■■■■■

```
# ■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■ PIXORA FACE
class Settings(BaseSettings):
    """■■■■■■■■■■■■■ ■■■■■■■■ ■■■■■■■■■■■■■■ ■■■■"""

    # ■■■■■■■■■■ ■■■■■ ■■■■■■■ (■■■■■■■■■■ ■■■■■■■ ■■■■■■■■■ ■■ ■■■■■■■■■)
    DATABASE_URL: str = "postgresql://user:password@localhost:5432/facepass_db"
    VECTOR_DATABASE_URL: str = "postgresql://user:password@localhost:5432/vector_db"
    PIXORA_DATABASE_URL: str = "postgresql://user:password@host:5432/pixora_db"

    # ■■■■■■■■■■ S3 ■■■■■■■■■■ (■■■■■■■■■■ ■■■■■■ ■■■■■■■■■)
    S3_ACCESS_KEY: str = "your_s3_access_key"
    S3_SECRET_KEY: str = "your_s3_secret_key"
    S3_BUCKET_NAME: str = "pixora-photos-bucket"
    S3_ENDPOINT_URL: str = "https://s3.example.com"

    # ■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■■■■■ ■■■■
    FACE_SIMILARITY_THRESHOLD: float = 0.5  # ■■■■■■ ■■■■■■■■■ (50%)
    MAX_FACES_PER_PHOTO: int = 10           # ■■■■■■■■■ ■■■■ ■■ ■■■■■
    EMBEDDING_DIMENSION: int = 512          # ■■■■■■■■■■■■ ■■■■■■■■ InsightFace

    # ■■■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■
    MAX_PHOTOS_PER_SESSION: int = 1000      # ■■■■■■ ■■■■■ ■ ■■■■■■■
    SEARCH_RESULT_LIMIT: int = 50           # ■■■■■■ ■■■■■■■■■■■■ ■■■■■■■
    INDEXING_BATCH_SIZE: int = 10           # ■■■■■■■ ■■■■■■■ ■■■■■■■■■■■
```

■■■■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■■

■■■■■■■■■■■ ■■■■■■■■■■■■ ■■■■■■■■ PIXORA FACE:

1. ■■■■■■■■■■■■■ ■■■■■■■■■■■■■■■ ■■■■■■■■■■ - ■■■■■■■■■■■■ ■■■■■■■■■■■■■■■■ ■■■■ ■■■■ ■■■■■■■■■■■■, ■■■■ ■ ■■■■ ■■■■■■■■ ■■■■ ■■■■■■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■■■ ■■■■■■■■■■

2. ■■■■■■■■■■■ ■■■■■■■■■■■■■ ■■■■ ■■■■■■■■ - ■■■■■■■■■■■■■■■■ ■■■■■■■■■■■ ■■■■■■■■■■■ ■■■ ■ pgvector ■■■■ ■■■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■ ■ ■■■■■■■■■■ ■■■ ■■■■ ■■■■■■■■■■■■

3. ■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■ ■ ■■■■■■■■■■■ ■■■■■■■■■■■■ - ■■■■■■■■■ ■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■ ■■■■■■ ■■■■■■■■■■■■, ■■■■■■■■■ ■■■■■■■■■■■ ■■■■■■■■■■■

4. ■■■■■■■■■■■■ ■■■■■■ ■■■■■■■■■■ - ■■■■■■■■■■■■■■■ ■■■■■■ ■■ 0.5 ■■ 0.7 ■ ■■■■■■■■■■■■■ ■■ ■■■■■■■■■■■■ ■■■■■■■■■■

5. ■■■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■ - ■■■■■■ ■■■■■■■■■■■ ■■■■■■■■■ ■■■■■■■■■■■■ ■■■■■■■■■■■■ ■■■■ ■■■■■■■■■■■■■■ ■■■■■■■■■■■■■■

■■■■■■■■■■■■■■■ ■■■■■■■■:

• **■■■■■■■■■■■ ■■■■■■■■■**: similarity = (A·B) / (||A|| × ||B||)

• ■■■■■■■■■■■■■: normalized_vector = vector / ||vector||

• ■■■■■: 1 - cosine_distance >= threshold

■■■■■■■■■■*:* ■■■ ■■■■■■■■■■ ■■■■■■*,* ■■■■■■■ ■ *IP*-■■■■■■■ ■ ■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■ ■■ ■■■■■■■■■ ■ ■■■■■■■■■■■■■ ■ ■■■■■■■■■■■■■ ■■■■■■■■■■■■■*.*