html

Comprehensive Completion Report: Months 1 & 2 — SelmApp

Performance Analysis, Processes, and App Flows

Report date:11 June 2025

Report version:2.0

Reporting period:

Overall status: ✅ **Successful and on schedule**

📋 Executive Summary

SelmApp Successful**100%**. infrastructure system authentication User management content Complete . .

🎯 Month 1: Infrastructure

Month 1 — Performance Summary

Section	Completion	Status
Backend & Infrastructure	100%	✅ **Complete**
Flutter foundation	100%	✅ **Complete**

Setup & documentation	100%	✅ Complete
Total	**100%**	✅ **Successful**

🏗️ Section : Backend & Infrastructure

✅ Key achievements:

- **FastAPI framework setup**
  - Project structure configuration
  - Middleware & CORS configuration
  - Error handling implementation
  - OpenAPI documentation
- **Database design & implementation**
  - PostgreSQL database setup
  - SQLAlchemy ORM configuration
  - 15 table
  - Migration system with Alembic
- **Authentication system**
  - JWT token authentication
  - Password hashing bcrypt
  - Session management
  - Security middleware

📊 Database tables (15 table):

User & Auth tables:

Table	Description	Main fields
users	user system	15
user_progress	Overall progress user	10
daily_progress	Daily progress user	8

Content & Learning tables:

Table	Description	Main fields
contents	content	12
vocabulary	vocabulary	10
grammar		9

Exercises & Quiz tables:

Table	Description	Main fields
exercises	exercises	11
exercise_attempts		9
quizzes	assessment	10
quiz_attempts		8
quiz_exercises	exercises	7

tables Achievements :

Table	Description	Main fields
achievements	Achievements	8
user_achievements	Achievements user	6
study_sessions	user	9
learning_goals		7

🔗 API Endpoints (18 endpoint):

Authentication Endpoints

Method	Endpoint	Description
POST	`/api/v1/auth/register`	User registration
POST	`/api/v1/auth/login`	User login

POST	`/api/v1/auth/refresh`	Refresh token
POST	`/api/v1/auth/logout`	User logout
POST	`/api/v1/auth/forgot-password`	Forgot password

User Management Endpoints

Method	Endpoint	Description
GET	`/api/v1/users/me`	User info
PUT	`/api/v1/users/me`	Update profile
GET	`/api/v1/users/profile`	Full profile

Content Basic Endpoints

Method	Endpoint	Description
GET	`/api/v1/content/`	Content list
GET	`/api/v1/content/levels`	CEFR levels
GET	`/api/v1/content/vocabulary/`	Basic vocabulary
GET	`/api/v1/content/grammar/`	Basic grammar

Exercise Basic Endpoints

Method	Endpoint	Description
GET	`/api/v1/exercises/`	Exercise list
GET	`/api/v1/exercises/{id}`	Exercise details
POST	`/api/v1/exercises/submit`	Submit answer

Progress Tracking Endpoints

Method	Endpoint	Description
GET	`/api/v1/progress/`	Overall progress

GET	`/api/v1/progress/daily`	Daily progress
GET	`/api/v1/progress/achievements`	Achievements

🔄 Key System Flows:

1. Authentication Flow

authentication OAuth2 files .

Authentication Process Diagram

Authentication Flow Diagram

2. Data Management Flow

Data Management Process Diagram

Data Management Flow Diagram

3. API Security Flow

API Security Process Diagram

API Security Flow Diagram

🎯 Month 2: User & Content Management

Month 2 — Performance Summary

Section	Completion	Status
User management	100%	✅ **Complete**

Content management	100%	✅ Complete
Total	**100%**	✅ **Successful**

👤 Section : User management

✅ Key achievements:

- **User profile system**
  - Personal information management
  - Learning settings
  - User preferences
  - Profile picture upload
- **Progress tracking system**
  - Score calculation
  - Study time tracking
  - Performance statistics
  - Strengths & weaknesses analysis
- **Level assessment system**
  - CEFR placement test
  - Level calculation algorithm
  - Personalized recommendations
  - Custom learning path

🗄️ tables (6 table):

Advanced authentication tables:

Table	Description	Main fields
oauth2_accounts	OAuth2 accounts (Google, GitHub, Facebook)	12

Personalization & learning tables:

Table	Description	Main fields

user_learning_profiles	Personal learning profiles	10
personalized_learning_paths	Personalized learning path	9
learning_path_milestones	Learning path milestones & goals	8

Recommendation & analytics tables:

Table	Description	Main fields
content_recommendations	Personalized content recommendations	11
learning_analytics	Learning analytics & statistics	13

🔗 API Endpoints (15 endpoint):

Advanced User Management

Method	Endpoint	Description
PUT	`/api/v1/users/preferences`	User preferences
GET	`/api/v1/users/statistics`	User statistics
POST	`/api/v1/users/avatar`	Profile picture upload
DELETE	`/api/v1/users/me`	Delete user account

OAuth2 Authentication

Method	Endpoint	Description
GET	`/api/v1/auth/oauth/{provider}/authorize`	OAuth2 start
POST	`/api/v1/auth/oauth/{provider}/callback`	OAuth2 callback
POST	`/api/v1/auth/oauth/login`	OAuth2 login

Progress Tracking

Method	Endpoint	Description

GET	`/api/v1/progress/analytics`	Progress analytics
POST	`/api/v1/progress/session/start`	Start study session
POST	`/api/v1/progress/session/end`	End study session
GET	`/api/v1/progress/streak`	Study streak days

Level Assessment

Method	Endpoint	Description
POST	`/api/v1/assessment/start`	level
POST	`/api/v1/assessment/submit`	Submit answer
GET	`/api/v1/assessment/result`	Placement test result
POST	`/api/v1/assessment/complete`	Complete assessment

📚 Section Section 2: Content management

✅ Key achievements:

- **Content structure based on CEFR**
  - Content organized into 6 levels
  - Topic categorization
  - Tagging and filtering
  - Advanced search system
- **Smart vocabulary system**
  - Spaced Repetition algorithm
  - Mastery level tracking
  - Total vocabulary
  - exercises vocabulary

🔄 Vocabulary Management Flow:

Vocabulary Management Flow Diagram

Vocabulary Management Flow Diagram

🔄 Content Recommendation Flow:

Content Recommendation Flow Diagram

Content Recommendation Flow Diagram

🔄 Core System Processes

1. User registration & onboarding process

User Registration Process

User Registration Flow Diagram

2. Study session process

Study session process

Study Session Flow Diagram

3. Content personalization process

Content personalization process

Content Personalization Flow Diagram

📊 Overall Project Stats

4,850

Backend lines of code

4,200+

Frontend lines of code

33

API Endpoints

21

Database tables

📊 Stats by month:

Month 1:

- **API Endpoints:**18 endpoint
- **Database tables:**15 table
- **Implemented areas:**Infrastructure, authentication, user management, basic content, exercises, progress

Month 2:

- **API Endpoints:**15 endpoint
- **Database tables:**6 table
- **Implemented areas:**OAuth2, advanced user management, progress analytics, level assessment, personalization

Grand total:

- **API Endpoints:**33 endpoint (18 + 15)
- **Database tables:**21 table (15 + 6)
- **:**9 SVG
- **system :**6 system

📊 Backend lines of code:

Section	Files	Avg. lines/file	Total lines
Models (21 table)	12 files	80 lines	960 lines
Schemas (33 endpoint)	10 files	60 lines	600 lines
CRUD Operations	12 files	120 lines	1,440 lines
API Endpoints (33 endpoint)	15 files	70 lines	1,050 lines
Services & Utils	8 files	90 lines	720 lines
Configuration & Core	6 files	40 lines	240 lines
Total	**63 files**	**-**	**5,010 lines**

Final estimate:4,850 lines (lines)

🔮 Preparation for Month 3

🎯 Readiness for Month 3 (Reading & Writing):

- ✅ **Content APIs**: Ready for Reading module
- ✅ **Exercise Framework**: Exercise framework ready
- ✅ **Progress Tracking**: Progress tracking system
- ✅ **User Interface**: Base UI components ready
- ✅ **Database Tables**: Required tables in place

📊 Conclusion — Months 1 & 2

SelmApp**Successful Complete**.

🎯 Achieved objectives:

- ✅ infrastructure Complete (100%)
- ✅ Authentication system advanced (100%)

- ✅ User management files (100%)
- ✅ Content and vocabulary system (100%)
- ✅ Flutter frontend foundation (100%) (UI not designed; initial setup completed)
- ✅ Analytics and personalization (100%)

🚀 Readiness for next steps:

( ) . system user .

Status : 🟢

🔐 OAuth2 Authentication Process Flow - SelmApp

🏗 Architecture Components

Backend Components

- **OAuth2Service**: Core service handling OAuth2 flows
- **Auth Router**: API endpoints for authentication
- **User CRUD**: User management operations
- **OAuth2 CRUD**: OAuth2 account management
- **Security Module**: JWT token handling

Database Models

- **User**: Main user account
- **OAuth2Account**: Linked OAuth2 provider accounts

🔄 Complete Authentication Flow

Complete OAuth2 Authentication Flow

OAuth2 Authentication Flow Diagram

🔄 Token Refresh Flow

Token Refresh Flow

Token Refresh Flow Diagram

🔧 Provider Configurations

Google OAuth2

```
{ "client_id": "GOOGLE_CLIENT_ID", "client_secret":
"GOOGLE_CLIENT_SECRET", "auth_url":
"https://accounts.google.com/o/oauth2/v2/auth", "token_url":
"https://oauth2.googleapis.com/token", "user_info_url":
"https://www.googleapis.com/oauth2/v2/userinfo", "scope":
"openid email profile" }
```

GitHub OAuth2

```
{ "client_id": "GITHUB_CLIENT_ID", "client_secret":
"GITHUB_CLIENT_SECRET", "auth_url":
"https://github.com/login/oauth/authorize", "token_url":
"https://github.com/login/oauth/access_token",
"user_info_url": "https://api.github.com/user", "scope":
"user:email" }
```

Facebook OAuth2

```
{ "client_id": "FACEBOOK_CLIENT_ID", "client_secret":
"FACEBOOK_CLIENT_SECRET", "auth_url":
```

```
"https://www.facebook.com/v18.0/dialog/oauth", "token_url":
"https://graph.facebook.com/v18.0/oauth/access_token",
"user_info_url": "https://graph.facebook.com/v18.0/me",
"scope": "email public_profile" }
```

📋 Detailed Step-by-Step Process

Step 1: Initialize OAuth2 Flow

```
GET /api/v1/auth/oauth/{provider}/authorize
```

Backend Process:

1. Validate provider (google, github, facebook)
2. Check provider configuration (client_id, client_secret)
3. Generate secure state parameter using`secrets.token_urlsafe(32)`
4. Create OAuth2 client with provider-specific configuration
5. Generate authorization URL with required scopes

Response:

```
{ "auth_url": "https://accounts.google.com/o/oauth2/v2/auth?
client_id=...&scope=openid+email+profile&state=...", "state":
"secure_random_string" }
```

Step 2: User Authorization

Frontend Process:

1. Store state parameter for validation
2. Redirect user to authorization URL
3. User authenticates with provider
4. Provider redirects back with authorization code

Step 3: Handle OAuth2 Callback

```
POST /api/v1/auth/oauth/{provider}/callback?
code=AUTH_CODE&state=STATE
```

Backend Process:

1. Validate state parameter (security check)
2. Exchange authorization code for access token
3. Retrieve user information from provider
4. Process user account creation/linking

🔧 Database Schema

Users Table

```
CREATE TABLE users ( id SERIAL PRIMARY KEY, email
VARCHAR(255) UNIQUE NOT NULL, username VARCHAR(50) UNIQUE NOT
NULL, hashed_password VARCHAR(255) NULL, full_name
VARCHAR(100), avatar_url VARCHAR(500), has_password BOOLEAN
DEFAULT TRUE, is_verified BOOLEAN DEFAULT FALSE, is_active
BOOLEAN DEFAULT TRUE, created_at TIMESTAMP DEFAULT NOW(),
updated_at TIMESTAMP DEFAULT NOW(), last_login TIMESTAMP );
```

OAuth2 Accounts Table

```
CREATE TABLE oauth2_accounts ( id SERIAL PRIMARY KEY, user_id
INTEGER REFERENCES users(id) ON DELETE CASCADE, provider
VARCHAR(50) NOT NULL, provider_user_id VARCHAR(255) NOT NULL,
provider_email VARCHAR(255), provider_name VARCHAR(255),
provider_avatar_url VARCHAR(500), access_token VARCHAR(1000),
refresh_token VARCHAR(1000), token_expires_at TIMESTAMP,
created_at TIMESTAMP DEFAULT NOW(), updated_at TIMESTAMP
DEFAULT NOW(), UNIQUE(provider, provider_user_id) );
```

🛡️ Security Considerations

State Parameter

- Generated using`secrets.token_urlsafe(32)`
- Prevents CSRF attacks
- Validated on callback to ensure request authenticity

Token Storage

- Access tokens stored temporarily (can be encrypted)
- Refresh tokens stored securely
- Token expiration properly handled

User Verification

- OAuth2 users are automatically verified
- Email verification not required for OAuth2 accounts
- Provider-verified identity trusted

🎯 Provider-Specific Considerations

Google OAuth2

- Requires`openid email profile`scope
- Returns standardized user info
- Supports refresh tokens

🔍 Error Handling

Common Error Scenarios

1. **Invalid Provider**: Return 400 Bad Request
2. **Missing Configuration**: Return 500 Internal Server Error
3. **Invalid Authorization Code**: Return 400 Bad Request
4. **Provider API Failure**: Return 500 Internal Server Error
5. **User Creation Failure**: Return 500 Internal Server Error

Error Response Format

```
{ "detail": "OAuth2 authentication failed: Invalid
authorization code" }
```

Revision date:11 June 2025

Approval status: ✅