EU4 SaveStats

Project Development Plan Document

MSCS 621L Cloud Computing

Marist College

School of Computer Science and Mathematics

This design document is prepared for use by EU4 SaveStats (EUSS) development team. Its purpose is to provide guidance for the design, build, and test of the product, ultimately enabling a successful deployment for the end user.

Version	Author(s)	Description	Date
1.0	Frank Seelmann	Initial Version	Feb 26, 2025
1.1	Frank Seelmann	Created after successfully proving data flow	March 11, 2025

Project Overview

The goal of the EU4 SaveStats (EUSS) project is to create a web-based dashboard for viewing results from Europa Universalis IV (EU4) save game files. Users will be able to upload their save files to the service, which will then be processed to extract relevant data, which will be stored in a database, and displayed through a web interface. The file will also be saved after processing, so that the file can be re-processed if new features are added. Users will be able to add friends so that they can compare their save-game data. The project will utilize AWS services, including EC2 for processing, a database for storing extracted data, and S3 for storing the original save files.

Project Objectives

The following are the primary objectives of the EUSS project:

- Develop a system for users to upload Europa Universalis IV save files.
- Process save files with the help of an existing Rust-based parser (eu4save) library to extract the relevant data
- Store extracted data in a database for efficient querying.
- Implement a web dashboard to visualize key insights. For example:
    - Income and score over time
    - Average leader skills throughout the campaign so far
    - Total casualties (battle vs attrition)
- Utilize AWS services for scalability and reliability:
    - **EC2:** Process save files and extract relevant data.
    - **DB:** Store structured data for queries.
    - **S3:** Retain original save files for future reference.

- Allow users accounts
    - Have sign-up/log-in (with appropriate pages for each)
    - Let files be associated with a user
    - Allow users to add other users as friends to allow them to view each other's save file data

---

Project Scope

Europa Universalis 4 is a complicated grand strategy game. Consequently, there is a *ton* of data that can be gleaned from the save files. To complete the project on time, it is important to define the scope of the project ahead of time to avoid spending too much time developing the wrong aspects of the project.

- **In Scope:**
    - User authentication and file upload system.
    - Parsing and processing Europa Universalis IV save files.
    - Data storage and querying functionality.
    - Web-based dashboard for data visualization.
    - Integration with AWS (EC2, S3, Database).
    - Multiplayer game analytics.
- **Potentially in Scope**
    - Compressed save files ("Ironman" campaigns)
- **Out of Scope:**
    - Modding support or custom game rule tracking.
    - Advanced AI-based predictions or advice

Technologies & Tools

The following is an outline of the different programming languages, utilities, and libraries that will be used in the EUSS project:

- **Frontend:** Python – Flask Library (for dashboard UI)
- **Backend:** Rust (for save file processing), Node.js for potential decompression via "paperman" utility
    - Paperman is no longer in active development, so testing will need to be done to see if it properly handles the relevant information intended to be displayed for this project in the latest version of EU4
- **Database:** MySQL RDS Database, mounted to the EC2 instance
- **Cloud Services:** AWS EC2, S3, and RDS
- **File Processing:** eu4save Rust library, Paperman file decompression API
- **Development:** Docker, to compile the Rust program for AWS Linux on the Windows development computer. Compiling it in the EC2 instance takes uses more resources than is available in the free tier "micro" instance.

Project Timeline

The following is the current schedule for task completion. This document is intended to be updated if these dates change, or items fall into/out of scope.

Phase	Tasks	Estimated Completion
Research & Planning	Identify the key data points to collect and consider the packages/libraries required. Finalize DB schema	February 29

Backend Development I	Prove the established dataflow process in a local environment. Player-data only.	March 5
Backend Development II	Set up AWS environment – Compute, Database, and Storage. Prove the established dataflow process in the cloud environment. Minimal frontend functionality.	March 12
Backend Development III	Expand user functionality: Allow login via OAuth, and adding friends to view/compare against their save file data. Sign-up/Log-in pages	March 26
Frontend Development I	Expand frontend functionality/experience for file upload and adding/viewing friends. Begin development on leaderboard view	April 2
Frontend Development II	Have fully functional end-to-end experience for the end-user	April 9
Testing & Optimization	Try to break it and fix bugs. Dial back the very-open permissions	April 16
Expand Functionality	Add additional functionality, such as including data for all countries in a save file. Repeat tests to ensure no new breakages.	April 23
Project Completion	Final review and submission	April 30

Potential Challenges & Solutions

The following are some of the potential challenges to be expected during the development of EUSS:

- **Challenge:** Learning Rust for save file processing.
  Solution: Start with basic Rust tutorials and experiment with eu4save early in the process.

- **Challenge:** Ensuring data accuracy and completeness.
  Solution: Perform test runs with multiple save files and validate extracted data against known results. Ask friends to sign in and upload their save files as well, once that functionality is established
- **Challenge:** Time management. There are many aspects of this project that require attention
  Solution: As outlined in the Project Timeline, prioritize end-to-end functionality first, before expanding the feature-set. For example, it could be a good idea to leverage tools like ChatGPT or DeepSeek to create a serviceable front-end quickly, before taking the time to develop it further.

Success Criteria

If the following objectives are met, the project can be considered a success:

- Users can upload save files and receive processed data.
- Users can add friends, and cannot see the their save file data before doing so
- Extracted data is accurately stored and retrievable via queries.
- Web dashboard provides meaningful insights with a user-friendly interface. Users can see analytics for their own save files as well as those of their friends
- System operates reliably on AWS infrastructure.