University of Antwerp
Faculty of Science

Programming Project Databases

Assignment 2023–2024

University of Antwerp

Web-based Idle Game

Professor:
Prof. Bart Goethals

Supervisors:
Joey De Pauw
Marco Favier

Scrum Master:
Annick De Bruyn

February 14, 2024 ♡

1 Introduction

In the course *Programming Project Databases*, you will learn to develop an extensive software project in a team setting. This assignment is designed with the following learning objectives in mind:

- Build and use a database inside a software product.
- Work independently in a team.
- Find and learn about new technologies on your own.
- Plan work and distribute tasks.
- Think creatively and solve problems.
- Clearly report on progress and choices made.
- Write high-quality software with a focus on usability, efficiency, and extensibility.
- Deploy and maintain a software system in production.

This is a comprehensive project course. There is **no** written exam in June, and there is **no** retake in August. Your evaluation will be based on the delivered software product, taking into account the above objectives.

In addition to this introduction, you will find section 2 explains the basic requirements of this year's assignment, and section 3 where all practical matters and the evaluation process are explained in this document.

2 Assignment

The assignment is to create a web-based idle game along the lines of Grepolis, Tribal Wars, Travian and Clash of Clans.

In this type of game, the player controls a city with various buildings. Each building performs specific actions such as gathering resources, creating military units or trading and selling goods. Resources can then be spent to upgrade buildings and grow the city.

This year, the assignment is intentionally kept open-ended, with the idea of having each team give their own, personal spin on the general idea of a web-based game. We only require a few, quite generic functionalities to ensure the application is complex enough. Besides those, each team is free to fill in the assignment however they desire. Even if you want to propose an idea that is outside of the scope of these requirements, we are open to discuss whether the idea is challenging enough and would fit within the context of this course.

Each team has to submit a scope description by the end of the month (see section 3.7: Planning) that contains their plans for the application.

In the sections 2.1 and 2.2 more information about the basic requirements of the application in terms of entities and functionality is described. section 2.3 provides a list of technologies that can be used.

2.1 Entities

The following entities need to be present in your application.

Player
: The player controls the game. Players need to be able to interact with each other, for example through direct messaging or by forming a guild or by pinging with emojis.

Settlement
: A settlement can be a city, outpost or town for example. The player manages one or more settlements.

Building
: Settlements contain buildings that can perform certain actions and can be upgraded. For example a military building can produce soldiers and a mine can provide stone at a certain rate.

2.2 Basic Requirements

This section describes all basic requirements in terms of functionality. It is mandatory for each team to implement and demonstrate at least this functionality during the final evaluation to pass the course (see section 3.7: Planning). How you choose to support this functionality is up to you. We mainly want to see that the team has thought through the features to be implemented and that you can justify all the choices made.

Login
: Players can login with a username and password.

  For the basic requirements, *no* consideration is given to security. Therefore, it is not a problem if the web application is not completely secure against attacks. However, any measures taken to secure the website can earn additional points. Be sure to describe them in your report (e.g., password hashing with salt).

Map View
: One of the necessary complexities of the assignment is to work with spatial data. Settlements have to be positioned on a map where interactions with nearby settlements are possible, for example by attacking another player.

Multiplayer Interaction
: There is also a social aspect to the game. Players need to be able to form alliances and chat with each other, for example.

Background Updates and Cooldowns
: The game needs to have an idle nature, that keeps players coming back after a while to check on their progress and queue new actions. Certain actions need to take a while to complete (cooldowns), and over time, resources need to accumulate. Keep scalability in mind for this feature. For example it is not feasible to have an "update thread" running on the server for every user.

Cleanup & Consistency
: When an entity is deleted, the related entities should also be updated accordingly (cascade policy). This prevents the memory from becoming filled with data that

is no longer useful. For example, when a user is deleted, their associated personal information should also be deleted, however, some remnants of the player's activity may still remain depending on your game design.

Ensure a user-friendly web interface that is both intuitive and appealing. You can use a CSS library, as explained in section 2.3: Technology. Additionally, it is necessary to utilize an **A**pplication **P**rogramming **I**nterface (API) to implement some of the features. The API should adhere to the REST design principle, which includes, among other things, not maintaining state in the API and focusing on entities rather than operations.

The API provides a way to communicate directly with the server from the front end without having to refresh or reload the page, which can significantly enhance user-friendliness. In summary, carefully consider how you implement the features. Well-thought-out choices and designs can prevent a lot of unnecessary work. Always thoroughly justify these choices in your reports.

2.3 Technology

Inherently to this project, you will have to work with various new technologies. There is a wide range of tools, frameworks, libraries, and services that you must combine to implement the assignment. We suggest these "technologies" per category.

Do not let the choice of which technologies to use linger for too long. We recommend comparing some options as soon as possible and discussing with your team which ones you will proceed with. After the main choices are made, you can start reading documentation and tutorials to delve into the chosen technologies.

Webserver

- Flask https://flask.palletsprojects.com/
  The Flask framework in Python is recommended, but using a different programming language and/or webserver is also allowed.

Web Design

- HTML + CSS + Js
  These are the basic elements of every webpage.
- Front end framework
  Modern web pages are often written with a lot of logic on the client and clever use of asynchronous calls to create so-called "single-page applications". These frameworks support this design:
  - Vue.js https://vuejs.org/
  - React https://reactjs.org/
  - Angular https://angular.io/
- CSS Framework
  Using existing code for CSS can be useful, especially for mobile-first development. Here are some examples of popular CSS frameworks:

  - Bulma https://bulma.io/
  - Bootstrap https://getbootstrap.com/
  - Material https://material.io/

Database

- PostgreSQL
  The use of PostgreSQL is mandatory.
- Redis https://redis.io/
  Redis is an in-memory data store that can be used for caching values that need to be updated often.

Database Design

- DBdiagram https://dbdiagram.io/
  Online tool for drawing ER diagrams.
- DBdesigner https://www.dbdesigner.net/
  Same.

API

- JSON Web Tokens https://jwt.io/
  Standard for token-based claims. Can be used for stateless authentication.
- Apiary https://apiary.io/
  Online tool for documenting (and testing) APIs.

Assets

- itch https://itch.io/game-assets
  For various assets, mostly textures.
- Kenney https://kenney.nl/assets
  For various assets.
- incompetech https://incompetech.com/
  For music.
- sfxr https://www.drpetter.se/project_sfxr.html
  For sound effects.
- opengameart https://opengameart.org/
  For various assets.

Remember to respect licensing and give appropriate credit where needed!

Teamwork & Planning

- Git
  The use of a version control system is mandatory. These services often also offer, among other things, *projects* and *issues* for planning and organization. Once a

repository is created, give `@JoeyDP` read access. Choose from the following three services:

  - GitHub https://github.com/
  - Bitbucket https://bitbucket.org/
  - Gitlab https://gitlab.com/

- Miro https://miro.com/
  Miro is an "all-in-one workspace"that can be used for planning, organizing, and structuring. A handy tool for discussing and developing ideas.
- Jira https://www.atlassian.com/software/jira
  Agile development tool with support for backlog, sprints, roadmaps, and kanban boards. The use of Jira is mandatory (with a free license).

3 Practical

3.1 Teams

This project should be carried out in teams of 5 or 6 students. The teams were randomly assigned in advance and can be found on Blackboard.

3.2 Sprints

To manage the project, we adhere to the principles of Agile software development. A detailed introduction to this will be provided during the first class. Subsequently, your first *sprint* will begin. A sprint is a period of 2–3 weeks during which you work towards a specific goal. Annick De Bruyn will guide you, and at the end/beginning of each sprint, she will conduct a *scrum meeting* with your team. These meetings will take place in the meeting room M.G.023 next to the classroom (M.G.026).

Since a maximum of 4 meetings can be held per week, we have divided the teams into two groups: A and B, spread across the weeks. Teams with an odd number are in group A, and teams with an even number are in group B:

Group A: Teams 1, 3, 5, and 7
Group B: Teams 2, 4, 6, and 8

Refer to section 3.7: Planning for the schedule of when you are expected. In-person attendance at these meetings is mandatory for all team members and will be taken into account for the final score.

In addition to the comprehensive meeting at the end of a sprint, it is also advisable to meet in between to discuss things and collaborate. We recommend scheduling when you want to work on the project and starting those times with a short meeting called *daily* or *stand up*.

A useful tool to keep track and apply agile principles is Jira. Each team must create a project here and invite us so that we can closely follow your progress. Add the following users:

- annick.debruyn@hotmail.com
- bart.goethals@uantwerpen.be

When aligning the goals of the sprint, also consider the *Definition of Done (DOD)*. This is a checklist of conditions that all features must meet to be considered fully finished. Decide together what the DOD entails, and it must include at least the following:

- Code works (no bugs).
- Code is tested (automatic tests or manual with documented step-by-step scenarios).
- Documentation is completed (not just comments in code, but also a report).
- Feature has been reviewed by a team member.

3.3 Discord

In addition to meeting in person in a classroom, appointments can also take place on Discord. Especially for this course, a virtual classroom has been set up as an alternative. To gain access, use the following invite link: `https://discord.gg/ypAucmV2Bz`. This link takes you to the welcome page of the server, where you can find more practical information on how to use this virtual classroom.

It is recommended to register for a permanent account on Discord and install the desktop application. This way, you do not have to use the invite link every time and you can log in more smoothly.

3.4 Template

To help you get started, a template web application is available on Blackboard. Start by getting this code working locally for each team member separately (following the tutorial in the README). Afterward, you can deploy a version together on the production server (see section 3.5: Hosting).

3.5 Hosting

Exclusively for this course, a budget on the Google Cloud Platform is provided through Google's educational program. You can use this amount to rent a server per team for the duration of the course. It is, of course, not allowed to use this credit for personal purposes!

The specifics of how to use the Google Cloud Platform to host a web application will be explained during the practical sessions. To link you to the educational credit, one team member must provide a Gmail email address.

By the end of the first week, the given web application template must be working on the server (at least). This template can be incrementally expanded to perform the assignment. Use a Git repository (GitHub/GitLab/Bitbucket) with the *production* or *main* branch on the server. Furthermore, a DNS name will also be linked to your server in the form of: `team[x].ua-ppdb.me`.

A working version of your system must be running on the server at all times. Therefore, it is essential for each team member to set up a local development environment with a local test database. This allows you to implement new features smoothly and independently without affecting the production version.

Those interested can use Continuous Integration and Continuous Delivery tools such as Jenkins, CircleCI and Github Actions to automate this process.

3.6 Reporting & Presentation

Evaluation is done through three (interim) *reports* and *demos*. Do not underestimate the importance of documenting, and presenting! Functionality that we are not aware of cannot be assessed. Additionally, a cool idea is worthless if not thoroughly explained and justified.

Report

Technical documentation is part of the final product. It includes the design of the program as a whole, a database diagram with explanations, and a description of the functionality. In summary, it contains all technical information needed for the delivered system. Ensure that each team member is involved in the implementation, not just in project management or documentation.

Presentation

For presentations, we expect a working demonstration, where you receive feedback from the jury. A significant portion of the points will be based on the integration of the required functionality. Two interim presentations will be held during the semester, followed by a final presentation during the exam period of an online available web application.

For a demo, use your own laptop(s), so we strongly recommend checking everything thoroughly in advance (internet connection, connection to the projector, etc.) to ensure a smooth demo. Provide sufficient data and prepare a scenario in advance. While the report is for technical information, the demo should focus on functionality. Use the presentation mainly to demonstrate the features you are proud of. Slides are not necessary as long as the demonstration is clear. Make sure everyone participates in the presentations.

3.7 Planning

An overview of all important dates is given in Table 1 (weeks run from Wednesday to Wednesday). Unless stated otherwise, the deadline is 23:59.

For the *first milestone*, submit a scope description for your project on Blackboard. This includes a general description of the game along with a list of features that your team will implement. Make sure to list and explain all your design decisions as well as any other relevant information that has been decided in your team. On Wednesday 28/02/2024 each team briefly presents their scope description for feedback.

For the *second milestone*, you will be evaluated based on an online available demo of the application. We expect all decisions to be made regarding the design and database. Provide mockups (pages without a backend connection or drawings) for pages that are not yet implemented.

During the *final presentation*, we expect a live version of the complete finished product, possibly with your own extensions.

Tabel 1: Overview of planning and deadlines (red†).

Week	Date	Deadline/Planning
1	14/02/2024	Assignment and introduction to Google Cloud Platform
	20/02/2024†	Hello World application running on GCP
2	21/02/2024	Scrum theory, and planning Sprint 1 — teams A+B
	26/02/2024†	Scope Description A+B (milestone 1)
3	28/02/2024	Present Scope Description (milestone 1) Sprint 2 — teams A
4	06/03/2024	Sprint 2 — teams B
5	13/03/2024	Sprint 3 — teams A
6	20/03/2024	Sprint 3 — teams B
7	27/03/2024	Sprint 4 — teams A
8 9		Easter break
10	17/04/2024	Sprint 4 — teams B + milestone 2
11	24/04/2024	Sprint 5 — teams A + milestone 2
12	08/05/2024	Sprint 5 — teams B
13	15/05/2024	Sprint 6 — teams A
14	22/05/2024	Sprint 6 — teams B
16-	Exam	Final report (Blackboard, date depends on exam) Final presentation (See exam schedule for date)

3.8 Evaluation

In addition to functionality, you will also be assessed on the following criteria:

- Teamwork and planning. (4)
- Quality of reports and presentations. (2)
- Quality of program code. (4)
- Quality of the database design and SQL queries. (3)
- Quality, originality, and usefulness of the delivered features. (3)
- Quality of the demonstrations. (2)
- User-friendliness of the application. (2)

Although the course is called "Programming Project Databases", the focus is certainly not only on the database aspect but also on programming a large project and being able to work independently in a team.

With a fair distribution of tasks, and if everyone is capable of presenting their part and answering questions, all members of the group will receive the same points.

3.9 Contact

Every Wednesday from **9:00 to 12:00**, room **M.G.026** is reserved for meetings or working on the project together. Additionally, it is always possible to meet in the virtual classroom on Discord.

The teaching assistant is present (on Discord and in person) to discuss specific questions and issues related to the project. For urgent questions about the project, personal problems between team members, or technical issues, you can contact via email: joey.depauw@uantwerpen.be and marco.favier@uantwerpen.be.

Good luck!