Empires of Avatharia

Group 22

Christopher Raschke
Johan Wessman
Jashar Ghavampour
Johan Granberg

REQUIREMENTS DOCUMENT

1. Preface

1.1. Expected readership

This requirements document is in its current form expected to be read by the course administration for the course DD1363 at KTH, members of other teams taking the course and by developers of "Empires of Avatharia", group 22.

1.2. Version history

New versions of this document are to be versioned by the system "major.minor.revision".

- Major changes are larger changes to the product specification itself.
- Minor changes may be changes in layout and structure with a limited amount of changes to the product specification, as well clarifying sections to reduce ambiguity.
- Revision changes may be corrections of language, images and other smaller errors with minimal, if any, impact on the product specifications.

Version	**Date**	**List of changes**
1.0.0	January 4, 2008	-

2. Introduction

2.1. Who are the users and what problem does the system solve for them?

Empires of Avatharia is an online strategy game that can be easily accessed from any web browser anywhere in the world. The game doesn't solve any specific problem (since it is a game). It is based upon a set of unique game rules that have never been used in a game before. It does however contain elements that are very common in most games, but the mixture of these elements is unique to this game. That is why this game will be of real value to the user; because of its unique game play.
The users could be of any gender and any age, but it will most probably attract strategy interested males between the ages of 15 to 30. We decided to aim specifically for that age group since it is common knowledge that this is the group with the most "gamers" in it. Even though this game is going to be easy to learn we hope that it could produce scenarios that even the more "serious gamers" will find challenging.

2.2. The main uses of the system.

The main use of our system is a game that is meant to be played for fun. It is a strategic game played in any standard web browser making it very flexible to use on most computers with an internet connection. Here are some user scenarios that are meant to illustrate how different users might experience "Empires of Avatharia".

Case 1:

Stig, a twenty-five year old computer interested man, is currently visiting his grandmother over the weekend. He has not brought any computer with him and has therefore no access to any of the games he usually plays. His grandmother's computer is old and slow and can't handle any new games. Fortunately, Stig knows of the game called "Empires of Avatharia", where he is a member. He decides to log in and play some before the family dinner. When logging in he resumes control over his village and army, making new decisions in the game. When he is done he can log out which saves his progress to the server. Stig likes to play "Empires of Avatharia" because it is easily accessible and because it is possible to play as much or as little as one may want.

Case 2:

Alexander, a seventeen-year-old boy, is currently on a break between classes at school. He has nothing to do and decides to play some "Empires of Avatharia", which he is playing on his free time. The game is easily accessible from the schools computers. He logs in on the website and resumes control over his village, reading a battle report and sending his army on a new attack. After a while his break is over and he logs out and makes his way back to the classroom. He likes to play "Empires of Avatharia" because of the easy access, and being able to play it on short breaks at school. A lot of his friends also play the game which makes it that more fun for him.

Case 3:

Anders is at work and working on his computer. After working a while he decides to take a short break. On his work computer he can easily log in and play "Empires of Avatharia". He likes to play "Empires of Avatharia" because of the easy access and he thinks it is a perfect way of taking a break from work. When he is logged in he manages his village, reads some old battle reports and moves his army. When he is done he logs out, and saves the progress to the server which then takes over control, and he returns to work.

2.3. The context/environment in which the system is to be used.

The game isn't designed for any specific environment, as we made clear in the section above. It could be used in a home environment at any time, a school environment at breaks between classes or even at a work-related environment when the employees are bored or find themselves out of work. It's a very flexible game and therefore the environments in which it is used tend to reflect that flexibility.

The game doesn't require any state-of-the-art hardware or any complicated installation. It should be easy to access and shouldn't require anything else than an internet connected computer with a modern web browser. This includes most PC and MAC computers. The fact that the servers will be up and running "twenty-four seven" also adds to the flexibility of the environment in which the game could be used.

2.4. The scope of the system.

This is our In/Out list where we point out what we want the project to involve. The list may be changed during the course of the project if new questions arise as the work goes along.

Topic	In	Out
HTML based graphics	X	
Real-time strategy	X	
Login and account management	X	
Communications system between users	X	
Platform independence	X	
Animated graphics		X
3D Graphics		X
2D Graphics	X	
Requires constant presence by user		X
Only the user himself has access to his virtual assets	X	
Computer controlled players		X

Possibility to add AI in the future	X	
Music		X
Game sound		X
Limited number of players	X	
Multiple languages		X
Supervision by administrators		X
Database driven transactions	X	
All calculations are made on the server	X	
JavaScript for dynamic handling	X	
Ranking system	X	

2.5. The main factors that need to be taken in to account when designing and building the system.

One thing that may give us a lot of problems is “unreliable users”. Many of the users in our game are likely to be members of multiple online communities and will therefore have many account names and passwords to remember. The more passwords and account names you use the more likely are you to forget one. Therefore we need a system that can handle that problem without compromising game-account security. This could be done in a few different ways but one solution is to bind an account to an email address.
Another problem that may arise when dealing with unreliable users is the scenario where users are trying to exploit the game. Either they try outright cheating/abusing bugs or just using actions that makes the game imbalanced. “In-game”-imbalance could be easy handled through patching and monitoring the “in-game-environment”. Cheating/abusing bugs on the other hand is harder to handle. We need too make sure that all the user can do is ask the server questions and ask it to do things for them. If we let the users do anything that might affect others on their computers we’re in for a lot of trouble. There are also the odd cases of online-vandals that are just out to shut down our game and cause us as much trouble as possible. That need to be addressed with smart database handling methods, to make sure no information is lost even if we have a server crash or something like that. Two ways of fixing this problem is using logs for everything that happens to the server, that way we could just restart the server and update everything that needs updating according to our logs. Another way to do it could be to run parallel database servers.

Another thing to keep in mind is that everybody is not going to start playing our game at the same time, and thus we need to allow people to be able to have a decent chance at growing even if they didn’t join in when the server started.

2.6. Technologies and Risks

With the game being web based, our choices are narrowed down to a handful of technologies, given that we want to stay with the ones that have proven their worth on a large scale over the past few years. The most obvious available choices under these criteria are PHP, Java or .NET, with MySQL or PostgreSQL databases. They are all nowadays cross-platform. Since the game is web based it will more or less be necessary to also include (X)HTML, CSS and JavaScript.

2.6.1 Software

Most members of our group have enough knowledge of and experience with Java to work on the backend, and since Java is an established, reasonably stable, platform independent and widespread language, it is our main language of choice for this project. The mentioned attributes facilitate teamwork and enable us to distribute tasks efficiently, while reducing the risk of a total halt due to any long-term absence of a single member.

The Java Servlet API will be used for this, and that means that the front-end language will be Java Server Pages, JSP. The familiarity in the group with Servlets and JSP is not extensive, but the time to acquire enough knowledge to be able to do something useful is a matter of hours if the Java skills are good enough.

Most of the HTML and JavaScript skills needed for the web interface are concentrated to one or two members of the group, but the simple nature of these languages, along with the limited complexity of the interface, allow for all members to work with the interface using the many web resources available, albeit more time-consuming than for the more experienced developers. A great risk here are the differences in rendering and standards compliancy between the browsers, and much testing will be necessary in order to ensure that the game is playable at least in the most common browsers.

We will use the PostgreSQL database as most of us have experience with it, and for this project it will have more or less the same risks as described for Java above.

2.6.2 Hardware and system

The application itself doesn’t pose any demanding requirements on the hardware on neither the client nor the server side, but with a growing number of users there will be a heavier load in terms of disc access and memory utilization on the server side, which will require upgrades of existing servers, expansion to new servers, load balancing and so on. A modern home computer will accommodate at least a few hundred simultaneous users with ease so we will code with scaling in mind, but hardware upgrades are probably no concern until the number of users grows far beyond our group. The natural stops in the game allow for certain downtime, allowing maintenance and upgrades without a heavy impact on the user experience.

We will probably not have different servers for development, testing or staging while the

game is under development, this is for practical and financial reasons. We will try to use some revision control system such as Subversion to keep the code gathered, accessible, updated and structured. This is all likely to be done on the same computer, perhaps with a web hosting company or somehow through KTH.

2.6.3 Other risks

Even with its limited complexity, the greatest concern is still the web interface coding and the Servlet behind it. All members will have to acquire some new skills to be able to work efficiently as a team, and this may take too much time if it is not well coordinated from the start.

Using the standard, well tested frameworks, we have much less to worry about when it comes to stability, but there are still concerns and it's important to patch everything as soon as possible to avoid security issues with newly discovered vulnerabilities. This may be difficult when not having complete control of the hardware, as is the situation with web hosts, a very important fact to keep in mind if any expansion of the game is needed.

The infrastructure around the server is also important to keep in mind if things get more serious – a real fireproof server hall with redundant electricity and physically separated connections to the net is much better than having a server standing in the basement of some office.

Backups are crucial for us right from the start and we have to make sure we have updated copies of everything on the server in case something unexpected would happen, especially in the development phase.

3. Glossary

3. Explanations of frequently used words and expressions.

Gold and Mana:

The Gold and Mana attributes are basically two separate values which will increase every hour depending on the player's province setup. Players use resources to build buildings and/or to train units.

Province:

A province represents an online town. A province contains buildings and units. A province also contains (some) of the players Gold and Mana.

Building:

A sort of resource that is bound to a specific province. Buildings are used to increase the flow of "Gold and Mana" and also to enable the training of units.

Unit:

A sort of resource that is bound to a specific player. Units are used to increase the chances of winning a "war". High-level units and specific combinations of different types of units are the key elements of winning a "war".

Unit type:

A unit is exactly one of the following types: Archer, Heavy infantry, Pikeman, Cavalry or flyer. These unit types have different advantages and weaknesses against each other.

Level or lvl:

Each building and unit has a level. The level is an integer between 1 and 100. The higher level a building or a unit is the more valuable it is to the player. Depending on what building it is it will produce more benefits for the player if it is at a higher level. Units with higher level will perform better in average at a "War".

Upgrading:

Any building or unit that has not yet reached its maximum level may be upgraded to a higher level one. This replaces the existing unit or building.

Constructing a building:

A building takes time to build and upgrade. The time it takes to build or upgrade a building is calculated using a strictly ascending function based on the buildings level. This means that it takes longer to upgrade a high level building than a low level building.

Training:

A unit takes time to train or upgrade. The time it takes to train or upgrade a unit is calculated using a strictly ascending function based on the unit's level. This means that it takes longer to upgrade a high level unit than a low level unit.

GUI:

'Graphical User Interface' – it is the actual menus and framework visible to the user while he/she is using the program.

SUD:

System under development.

4. User requirements definition

4.1 Functional Requirements

When grading how critical a requirement is we use the following scale:

1. **Not important**: The system will not suffer. Implementing it however will make the system more enjoyable.
2. **Important**: Users may take advantage of weaknesses in the game.
3. **Critical**: The game will suffer deeply if this is not implemented.
4. **Very Critical**: The game will not function at all if this requirement is not met.

The error handling is crucial, these "sub-requirements" vary from being "Important" (2) to "Critical" (3).

4.1.1 – Registering

Rationale: *When the user first comes in contact "with Empires of Avatharia" he/she will probably need to register him- or herself as a new user on the "Empires of Avatharia" webpage. All the users achievements are then stored in his or her user account. The accounts will then interact with each other as described in the sections below.*

Dependencies: None

Importance: 4 "Very critical"

Function: Anyone who wants to play "Empires of Avatharia" has to register himself. Registering is the only way to create a user account, which the player uses to store all his achievements in the game.

Input: The registering form requires the following input from the user;

- His or her full name
- A unique username
- A unique email-address
- Desired password
- Desired password (again).
- In-game choices;
    - In-game faction
    - In-game start location

Output: A user account stored in the database that is accessible from the "Empires of Avatharia" webpage.

Anticipated errors:

- **The username is not unique [1], to short or to long:** The user is prompted to input a different username.
- **The password is to short or contains "illegal" characters:** The user is prompted to input a "valid password" [2].
- **The user mistypes his or her password:** The user is prompted to input two identical [3] passwords.
- **The email address is invalid [4]:** The user will have to try to register again. No user account is created.
- **The in-game start location is invalid [5].** The user is prompted to input a new start location.

Explanations / Clarifications:
[1] The username is already used by another player.
[2] The password must be at least 5 characters long and has to consist of at least one number. It can consist of any of the characters A-Z and/or any of the numbers 0-9. It can be no longer than 20 characters.
[3] The same password has to be typed in twice. These passwords are case-sensitive.
[4] It does not belong to the user or is not an email address at all.
[5] If the spot that the user selected belongs to another player or if it collides with his previous "in-game faction" choice then the start location is invalid.

4.1.2 – Logging in

Rationale: *In order to be able to play the game or access any other of the "Empires of Avatharia" material, the user has to identify him- or herself by "logging in".*

Dependencies: The user has to be registered (See 1.1 Registration)

Importance: 4 "Very critical"

Function: In order to access any in-game material the user has to log in to his user account.

Input: The user has to supply a valid username along with its corresponding password.

Output: The user is considered "logged in" by the system and has access privileges to in-game resources [1] that belong to the user.

Anticipated errors:

- **The username could not be found in the system database:** The user is prompted to input a valid username and password.
- **The password is does not correspond to the username:** The user is prompted to input a valid username and password.

Explanations / Clarifications:
[1] Resources include buildings, units, armies, gold and mana.

4.1.3 – Recover password

Rationale: *A user may lose his password. The recover password function is a safety precaution meant to protect the user.*

Dependencies: The user has to be registered – *See 1.1 Registration*

Importance: 1 "Not important"

Function: The user will be able to recover his or her password.

Input: Username

Output: The user will be receiving his or her password by email.

Anticipated errors:

- **The username could not be found in the system database:** The user is prompted to input a valid username and password.
- **A user requests his passwords a second time within 2 hours [1] of the first time:** The request is discarded and the user is alerted.

Explanations / Clarifications:
[1] Multiple requests may be used to "spam" users with password recoveries.

4.1.4 – The in-game-environment

Rationale: *The in-game environment is accessible when the user has logged in and represents the actual game. It is here that the user makes all his choices-choices. This is the core of the system.*

Dependencies:

1. The user has to be registered – *See 1.1 Registration*
2. The user has to be logged in – *See 1.2 Logging in*

Importance: 4 "Very Critical"

Function: The user will be able to make in-game choices which include:

- Build buildings
- Train units
- Chat with other players
- Send private messages to other players
- Manage armies
- View:
    i. "the province list"
    ii. "the map"
    iii. a "combat log"
    iv. the current "ranking"

Explanations / Clarifications:
Players: A user is considered a player if and only if he or she is registered.

Input/Output: In order to define the in- and output of the system, each item in the function list has to be considered as a requirement. These items are too different to be generalized.

4.1.5 – Build buildings

Rationale: *By using in-game resources such as "gold" and "mana"[1] the user can build buildings in his "provinces" [2]. Buildings provide different benefits in the province that they are built. Certain buildings enables training of units (see below).*

Dependencies:

1. The user has to be registered – *See 1.1 Registration*
2. The user has to be logged in – *See 1.2 Logging in*

Importance: 4 "Very Critical"

Function: The user will be able to build a building [3] or upgrade a building [4] by spending some (or all) of his gold and/or mana [1] in the corresponding province [2].

Input: The building that the user wants to build (or upgrade).

Output: The building is added to the corresponding province [2] after a specific amount of time [6]. The system adds/replaces the building to the user's current in-game resources. Resources are subtracted from the users province's total gold and mana.

Anticipated errors:

- **The province does not contain enough resources to build the building:** The user is alerted that this is the case. No building is built and no resources are subtracted from the province.
- **The building has reached its maximum level [5]:** The option to upgrade will be grayed out (not available) to the user.

Explanations / Clarifications:
[1] The Gold and Mana attributes are basically two separate integers which will increase every hour depending on the player's province setup. Players use resources to build buildings and/or to train units.
[2] A province represents an online town. A province contains buildings and units. A province also contains (some) of the players "Gold" and "Mana" [1].
[3] A resource that is bound to a specific province [2].
[4] Any building [3] may be upgraded to a higher level [5] one. This replaces the existing building.
[5] Each building and unit has a level. The level is an integer between 1 and 100. The higher level a building is the more valuable it is to the player. Depending on what building it is it will produce more benefits for the player if it is at a higher level.
[6] A building takes time to build or upgrade. The time it takes to build or upgrade a building is calculated using a strictly ascending function based on the buildings level [5]. This means that it takes longer to upgrade a high level building than a low level building.

4.1.6 – Train units

Rationale: *A unit is the most common way of interaction between players (other than chatting and using the private message sending system). Units can be organized in armies and sent to other players to "pillage" or even to capture an enemy province. The training and upgrading of units is an essential part of the game.*

Dependencies:

1. The user has to be registered – *See 1.1 Registration*
2. The user has to be logged in – *See 1.2 Logging in*

Importance: 4 "Very Critical"

Function: The user will be able to train a unit [3] or upgrade [4] it by spending some (or all) of his gold and/or mana [1] in the corresponding province [2].

Input: The unit that the user wants to train (or upgrade).

Output: The unit is added to the corresponding province [2] after a specific amount of time [6]. The system adds/replaces the unit to the user's current in-game resources. Resources are subtracted from the users province's total gold and mana.

Anticipated errors:

- **The province does not contain enough resources to train the unit:** The user is alerted that this is the case. No unit is trained and no resources are subtracted from the province.
- **The unit has reached its maximum level [5]:** The option to upgrade will be grayed out (not available) to the user.

Explanations / Clarifications:
[1] The Gold and Mana attributes are basically two separate integers which will increase every hour depending on the player's province setup. Players use resources to build buildings and/or to train units.
[2] A province represents an online town. A province contains buildings and units. A province also contains (some) of the players "Gold" and "Mana" [1].
[3] A resource that is bound to an army [7], but can move between provinces [2].
[4] Any unit [3] may be upgraded to a higher level [5] one. This replaces the existing unit.
[5] Each unit has a level. The level is an integer between 1 and 100. The higher level a unit is the more valuable it is to the player because it has a greater chance of succeeding with missions [8] when used in an army.
[6] A unit takes time to train or upgrade. The time it takes to train or upgrade a unit is calculated using a strictly ascending function based on the unit's level [5]. This means that it takes longer to upgrade a high level unit than a low level unit.
[7] An army is a collection of units that all belongs to the same player. An army can be sent to do missions [8].
[8] A mission is an interaction between player's resources. Depending on if the mission succeeds or not the players involved will lose and/or gain resources. The most common

mission is “pillage” where a player sends his or her army to steal resources from another player (that belongs to an enemy faction [9]).
[9] Any faction that the player does not belong to.

4.1.7 – Chat with other players

Dependencies:

1. The user has to be registered – *See 1.1 Registration*
2. The user has to be logged in – *See 1.2 Logging in*

Importance: 1 “Not Important”

Function: Players will be able to chat with each other.

Input: The player inputs his chat [1] message.

Output: All other players that are connected to the same channel [2] receive the message.

Anticipated errors:

- **A user tries to send messages more often than once each other second [3]:** The user is alerted about this. No one in the channel receives the message.

Explanations / Clarifications:
[1] Instant messaging between users.
[2] Much like a room. Things that players “say” can only be “heard” if you are in the same room/channel.
[3] Limited to prevent spam/abuse.

4.1.8 – Send private messages to other players

Rationale: *The private messaging system works much like any mail program would, except that messages can only be sent within the “Empires of Avatharia” system. Players have their own inboxes and can send, receive and delete messages.*

Dependencies:

3. The user has to be registered – *See 1.1 Registration*
4. The user has to be logged in – *See 1.2 Logging in*

Importance: 1 “Not Important”

Function: Players will be able to send private messages to each other.

Input: The player inputs his message and the recipient’s username.

Output: The recipient receives the message.

Anticipated errors:

- **A user tries to send messages more often than once each other minute [1]:** The user is alerted about this. The recipient does not receive any message.

Explanations / Clarifications:
[1] Limited to prevent spam/abuse.

4.1.9 – Manage armies

Dependencies:
1. The user has to be registered – *See 1.1 Registration*
2. The user has to be logged in – *See 1.2 Logging in*
3. The user must have at least one unit in any province – *See 1.4.2 Train units*

Importance: 4 "Very Critical"

Function: Players will be able to move units [1] between armies [3] and send them to missions [4].

Input:
- ***If the player wants to move a unit:*** The unit in discussion. And the army he wants to move the unit to.
- ***If the player wants to send an army on a mission:*** The army in discussion and the mission he wants to send it on.

Output:
- ***If the player wants to move a unit:*** The system will move the selected unit from the army that it was originally in to the selected army.
- ***If the player wants to send an army on a mission:*** The army will be removed from the province [2] during the mission. Depending on the circumstances [6] they:
  - Return to the province. Resources are added to the player's province.
  - Not return at all.
  - Appear at a newly captured province.

Anticipated errors:
- **A player tries to send his army on a hostile mission towards a friendly province:** The user is alerted that this action is not possible. The army is not sent.

Explanations / Clarifications:
[1] A resource that is bound to an army [3], but can move between provinces [2].
[2] A province represents an online town. A province contains buildings and units. A province also contains (some) of the players "Gold" and "Mana".
[3] An army is a collection of units that all belongs to the same player. An army can be sent to do missions [4].
[4] A mission is an interaction between player's resources. Depending on if the mission succeeds or not the players involved will lose and/or gain resources. The most common mission is "pillage" where a player sends his or her army to steal resources from another player (that belongs to an enemy faction [5]).

[5] Any faction that the player does not belong to.
[6] In this case the army can have different missions and fail or succeed with those missions. Depending on these circumstances many different outcomes are possible. *(This is explained in greater detail in the "System requirements specification"-section)*

4.1.10 – Select a view

Rationale: *Different views represent different aspects of the game. These "views" are available so that the user can gain a deeper understanding of the mechanics behind the game.*

Importance: 3 "Critical"

Dependencies:

4. The user has to be registered – *See 1.1 Registration*
5. The user has to be logged in – *See 1.2 Logging in*

Function: The user has to be able to change between different views [1]. He has to be able to view:

- The province list
- The map
- Combat log
- The current ranking

Input: The view that the player wants to change to.

Output: The system changes the current view to the selected view.

Anticipated errors: None

Explanations / Clarifications:
[1] These different views and the elements surrounding them are discussed in the "System requirements specification"-section.

4.2 Non-functional requirements

There are several different types of non-functional requirements that our system must meet, though to different degrees. To make it clear we will for each type describe how and to what degree our system must meet the requirement.

4.2.1 – Performance Requirements

To begin with we will talk about the *performance requirement*. Our system will not make performance an issue at the user end, and therefore there is no real requirement on how much the system will use the user's computers resources.
On the other hand our system must meet a high performance requirement when handling the communication between users and server. The system must be efficient when sending information between user and server so there is no apparent delay while playing.

Having a limit to the maximum number of users that can be connected at a given time will also help the system be more efficient.

4.2.2 – Space Requirements

As no installation of the game is required on the user's computer, there is no great demand for free disc space on the player's computer, other than for the GUI data that the browser saves in its cache.

4.2.3 – Reliability Requirements

The third requirement type is *reliability*. Our system must have high reliability; it is highly undesirable to experience server crashes. The server computer needs to be reliable and our program must be written in a good way, and also tested so that crashes are avoided.
If a crash does occur, the system must stay consistent and information about what the users have done saved. This will most probably be solved by the system using a database with logs and backup as the only means to saving information, no files that must be rewritten etc.

4.2.4 – Portability Requirements

The next requirement is *portability* and where the system is to be used. The requirement in our case is that the game must be able to be played on most web browsers, (especially Internet Explorer, Firefox/Mozilla), but only on computers.
The game is not supported on other platforms than computers, even though it may work on a mobile phone with a web browser and the like.

4.2.5 – Usability Requirements

Usability is another type of requirement that we want the game to meet to some extent.
The game must be easy-to-use and almost like navigating any usual web page. This means that there will be no advanced interfaces and the like, which will make the game easy to play, even for people with internet-usage only and casual gamers.

4.2.6 – Privacy Requirements

Privacy is also something we want to include in the system. The privacy requirement is, for our system, that a user's information is not shown to the public. This includes information like the user's real name and e-mail address.

4.2.7 – Standards Requirements

The last requirement which our project involves is a *standards requirement* throughout the development of the system. All of the documentation and written work must be saved as

Microsoft Office .doc files. We have chosen this format to make sure that everyone in the group can open and read/change documents sent around before handing them in as PDF versions.

4.3 Use cases

4.3.1 Registering to the game

Goal: Make a registration to the game.

Primary actor: The user who wants to be able to play the game.
Preconditions: The user is connected to the "Empires of Avatharia" webpage.
Success guarantee: The user has created a unique user account in the SuD.
Stakeholders:

- **The user:** Wants to be able to register. Does not want to provide too much private information. Wants the registering to be swift and easy.
- **Owners of the SuD**: Requires that anyone will be able to register as a new user. The registering should be swift and easy so that it doesn't scare potential user.

Main success scenario:

1. The user enters the SuD's website.
2. The user chooses the registering option.
3. The system asks the user to input registering information.
4. User enters his name, desired username, email address, password and two in-game choices.
5. The system checks the information and sends a validation to the users email address. The information is stored in the newly created user account. The user is sent to the login screen.
6. The use case ends.

Extensions:
4a. One of the submitted fields have incorrect values entered.
4a1. The user receives a notification about what field was wrongfully entered.
4a2. The user supplies the information again but correct.
4b. The username and/or e-mail address already exists in the database.
4b1. The user is informed that the username and/or e-mail address is already linked to another account.
4b2. The user enters a new username and/or e-mail address.

4.3.2 To Login to the game

Goal: To log in into the game.

Primary actor: The user who wants to play the game.
Preconditions: The user is registered to the game and connected to the "Empires of Avatharia" webpage.

Success guarantee: The user is successfully logged in to the SuD and able to play the game.
Stakeholders:

- **The user:** Wants to be able to log in and play. Wants the login to be fast and easy.
- **The owners of the SuD:** Wants the potential users to be able to log in to the game in an easy and efficient fashion.

Main success scenario:

1. The user enters his/her username and password to login.
2. The system verifies that the password entered is the real one corresponding to the username specified and that there are no incorrect characters included.
3. The user is logged in and can play.

Extensions:
1a. The website is offline.
1a1. The user will receive a normal notification that the website is down.
3a. The username includes incorrect characters.
3a1. The website notifies the user about the username including an illegal character.
3a2. The user enters the username and password again, hopefully correct this time.
3a3. The system verifies that the password entered is the real one corresponding to the username specified.
3a4. The user is logged in and can play.
3b. The username entered is not associated with any account.
3b1. The system notifies the user that an account with the specified username doesn't exist.
3b2. The user tries to re-enter the username.
4a. The password entered was not correct.
4a1. The user receives a notification that the password was incorrect and is prompted to enter the correct one.
4a2. The user enters the password again and presses login.

4.3.3 Build a building

Goal: To start building a building.

Primary actor: Any registered player.
Preconditions: The user is logged in to his account.
Success guarantee: The user has build/upgraded a building.
Stakeholders: The user: Wants to build a building. It shouldn't be too hard to figure out what to do.

Main success scenario:

1. The system will display the in-game environment.
2. The user will select the "Province list".
3. The system will change the "main view" to the "Province list".
4. The user will select a province from the list.
5. The system will change the "main view" to the corresponding "province".
6. The user will select the "town centre" building.
7. The system will change the "main view" to the "town centre" and present a list with all the buildings that the user is able to build in this province.
8. The user will select one of these buildings and then click on the "buy"-button.

9. The system will process the request for buying this building.
10. The use case ends.

Extensions:
2-8a. The user might select another option.
2-8a1.The use case continues at 4 whenever the user selects the “province list”-tab or at 6 if the user selects the “Capitol”.
6a. The user may click on any other building.
6a1.The user case continues at 8 whenever the user selects the “town centre”.
8a. The user might have insufficient resources to build the selected building.
8a1. The building is not built.

Variations:
2. The user may choose the “Capitol” in which case the use case continues at 6.
4. The province can be any of the provinces in the game that the player owns.
7. The buildings can also be “upgrades”.

4.3.4 Train a unit

Goal: To start training a Unit

Primary actor: Any registered player.
Preconditions: The user is logged in to his account.
Success guarantee: The user has started training/upgrading a unit.
Stakeholders: The user: Wants to train a unit. It shouldn’t be too hard to figure out what to do.

1. The user will select the “Province list” option.
2. The system will change the “main view” to the “Province list”.
3. The user will select a province from the list.
4. The system will change the “main view” to the corresponding “province”.
5. The user will select the “town centre” building.
6. The system will change the “main view” to the “town centre” and present a list with all the units that the user is able to train or upgrade in this province.
7. The user will select one of these units and then click on the “buy/upgrade”-button.
8. The system will process the request for training or upgrading the unit. (The unit is trained)
9. The use case ends.

Extensions:
1-7a. The user might select another option.
1-7a1.The use case continues at 3 whenever the user selects the “province list” option or at 5 if the user selects the “Capitol”-option.
5a. The user may select any other building.
5a1.The user case continues at 7 whenever the user selects the “town centre”.
7a. The user might have insufficient resources to train the selected unit.
7a1. The unit is not trained.

Variations:
1. The user may choose to the “Capitol” option in which case the use case continues at 6.
3. The province can be any of the provinces in the game that the player owns.
7. Possible units are special cases of:
a) Archers
b) Cavalry
c) Pikemen
d) Heavy infantry
e) Flying

4.3.5 War

Goal: To send an army to war.
Primary actor: Any registered player.
Preconditions: The user is logged in to his account.
Success guarantee: The user has sent his army to war
Stakeholders: The user: Wants to send one of his armies to war.

Main success scenario:

1. The user will select the “Province list” option.
2. The system will change the “main view” to the “Province list”.
3. The user will select a province from the list.
4. The system will change the “main view” to the corresponding “province”.
5. The user will select his “offensive army”.
6. The user will then click on the “send to war button”.
7. The system will change the “main view” to the map.
8. The user will select an enemy province.
9. The system will display 3 options; Pillage, Attack, Abort.
10. The user selects one of these options.
11. The system will send the selected army to war.
12. The use case ends.

Extensions:
2-7a. The user might select another option.
2-7a1. The use case continues at 4 whenever the user selects the “province list” option or at 6 if the user selects the “Capitol”.
5a. The offensive army might be empty.
5a1. The army cannot be sent to war.
8a. The user might select an allies province or an empty square.
8a1. The operation is aborted.
9a The user may choose to abort.
9a1. The use case resets to 5.
9b.The province that the player has chosen in 9 might be an enemy capitol.
9b1. The Attack option is not available.

Variations:

2. The user may choose to select on the “Capitol” option in which case the use case continues at 6.
4. The province can be any of the provinces in the game that the player owns.
7. The province can be any other province that belongs to a player of another faction.

4.3.6 Private Message

Goal: To send a private message.

Primary actor: Any registered player
Preconditions: The user is logged in to his account.
Success guarantee: The user has sent a private message.
Stakeholders:

- The sending user: Wants to send a private message to someone.
- The receiving user: May want to receive a private message.

Main success scenario:

1. The user presses selects the “Chat” option.
2. The program displays the Chat window.
3. The user presses the “new message” button.
4. The program shows an empty mail.
5. The user inputs receiver name and a text message to the receiver.
6. The user selects “send”.
7. The system sends the private message to the receiver.
8. The use case ends

Extensions:
1-6a. The user might select another tab.
1-6a1.The use case continues at 3 whenever the user selects the “Chat”-tab
6a. No receiver name.
6a1. Nothing happens.
6b. No text message.
6b1. Nothing happens.

Variations:
5. Receiver is
a. A player in your faction.
b. A player in an enemy faction.
c. A server admin.

4.3.7 Chat message

Goal: Write a chat message.

Primary actor: Any registered player
Preconditions: The user is logged in to his account.
Success guarantee: The user has sent a chat message.

Stakeholders:

- The sending user: Wants to send a private message to someone.
- The receiving users: May want to be able to read what the sending user said.

Main success scenario:

1. The user selects the "Chat" option.
2. The program displays the Chat window.
3. The user chooses a channel by selecting a channel name from a list.
4. The program displays the channels chat window.
5. The user types a message.
6. The user sends the message by pressing "ENTER".
7. The system sends the message to all the people in the channel.
8. The use case ends

Extensions:
1-6a. The user might select another tab.
1-6a1The use case continues at 3 whenever the user selects the "Chat"-tab
6b. The user has already sent a message within the last two seconds.
6b1. The send is ignore. The user must wait an additional second before sending a new message.

Variations:
3. Channel name is
A String.
A Number.

4.3.8 Check "Ranking"

Goal: To view the total ranking

Primary actor: Any registered player
Precondition: The player is logged in.
Stakeholders and interests: Player – wants information about the status of different players, including the own status.
Success guarantee: The user has received a list of players with various scores.
Minimal guarantee: The user has received a list of the top players and their total scores.

Main success scenario:

1. The user will select the ranking section in the game menu.
2. The system will display a list of a cross-section of the game players, ordered by their total score, along with numbers that indicate the player's current scores in various categories.
3. The use case ends.

Extensions:
2a. The user selects to view a different part of the list with users with higher or lower scores than the current list.
2a1. The system displays another section of the player list.

4.3.9 View Map

Goal: To view the map

Primary actor: Any registered player
Precondition: The player is logged in.
Stakeholders and interests: Player – wants information about the virtual map.
Success guarantee: The user has through a map received desired information about the virtual world of the game in terms of distribution of territory between different players.
Minimal guarantee: The user has received information about his or her current status in the virtual world.

Main success scenario:

1. The user will select the map section in the game menu.
2. The system displays a map of the virtual territory distribution with selectable areas, together with map controls for viewing other sections of the map.
3. The user has received the desired information.
4. The use case ends.

Extensions
2a. The user scrolls the map.
2a1. The system displays another adjacent section of the map.
2b. The user selects an area of the map
2b1. The system displays information about the status of the selected area.

4.4 Standards

Process standards
We will use the IEEE 12207 standard as the foundation for the development process. This standard will serve as a guideline and we will not aim for strict compliance.

Web standards
"Empires of Avatharia" will strictly comply with the web language standards set by the The World Wide Web Consortium, W3C, for XHTML, JavaScript and CSS.

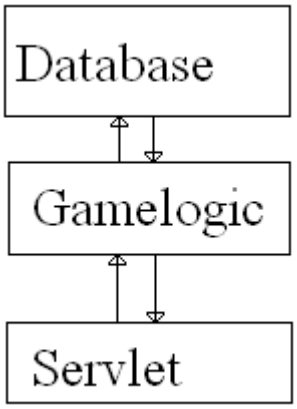
5. System architecture

5.1 High level overview of the system

When looking on our system architecture from a high level, we anticipate it to be divided into two main parts; the main program and the information database.

The main program will do all of the actual work in the system and it will handle all the processes going on. These include things like communicating with the information database, calculating battles, updating the world when needed and the like. The main program also sends out the information needed for the clients so that they can show the appropriate information in their respective web browsers.
So you can view the main program as being divided into three parts itself. One who communicates with the database, one that calculates and processes all the changes in the game world and one that sends out the information to the users.

The information database on the other hand does not hold any active part in the changes to the system. It is only used for storing all of the information used by the system.
This includes information about users, what villages the users own, what armies the users control, what buildings a village has and also what kind of troops an army consists of and can be trained.
As explained above, the database will not take any active part in the changes and processes of the system, but will only receive orders from the main program. These orders will tell how the database is supposed to change when new things are added to the game world or when things are supposed to be removed, after say a combat.



A basic picture of how the system will look.

5.2 Components that are reused

Our system is, as stated earlier, only made of two big parts; the main program and the information database.
The main program is, as explained earlier, divided into three parts. The parts that communicate with the users and with the database can be viewed as being reused by the system. Whenever a change occurs and something changes in the game new information must

be sent to the users and to the database. The third part on the other hand is not really reused but instead in use constantly. It is the heart of the program which runs constantly and calculates battles, changes to villages etc.

The database is the main component being reused a lot by the main program.
Every time the game world changes, (i.e. a new user registers, a building is built, a new unit is trained etc), the database must be updated. When such a thing occurs the main program sends a SQL-command to the database so that it updates the needed information.
Therefore the main program reuses the database very often, (fetching, updating, adding and removing information).

6. System requirements specification

6.1 Functional requirements specification

This section will provide you with more information about the already mentioned functional requirements. Any details that where left out in that section can be found here. Please note that the figures in this section are not meant as an actual sketch of the game. These figures are meant to help the reader get a mental picture of what we are actually talking about.

6.1.1 Registering

In-game factions

The user can choose to side with any of the 3 in-game factions. Depending on his or her choice he will then select an in-game starting location. The system will display a so that this choice can be done easily by clicking on the desired location of the map. The location is represented both graphically and by a game map coordinate system.
Every faction is considered an enemy to all members of another faction. A player cannot choose to start next to or adjacent to an enemy faction. When a player has chosen his starting location the area surrounding his province will now come under the control of the faction that he or she belongs to. This will prevent other players from choosing their starting location right next to that player. A player can never change his choice of faction.

6.1.2 The in-game environment

The views

The user may choose to change his survey of the game at any time by clicking on one of the many tabs. When a tab is selected, the main window changes to a window corresponding to the chosen tab. The tabs include;

- Capitol
- Province list
- Map
- Ranking

- Chat
- Combat Log

This window is by default set to the capitol. Here is a rough example of how the windows and tabs may be orientated:

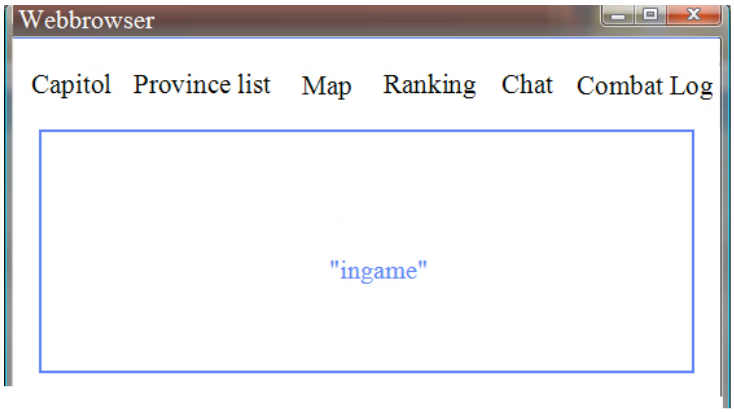


Fig 6.1.2.a

The Capitol view

When the user has processed the login he is then sent to an in-game environment. This environment is set to show the users "Capitol" by default. The capitol is basically the province that the user starts with. The capitol is bound to a specific player and can never be changed. The capitol view will be very similar to viewing any other province but because of the special properties of a capitol:

- They cannot be "captured" by other players (see Fighting).
- The only province that you start with.

Some choices that the user can make will differ.

The Province list view

When the user selects the "Province list"-tab, in his or her web browser, his main window is changed to a list, containing all his provinces. If the list is too long to fit in the window then scrolling is enabled. After each province name is a short line of information about that province. This information includes generated gold income and the average "lvl" of all the buildings in that province. Here is a rough example:

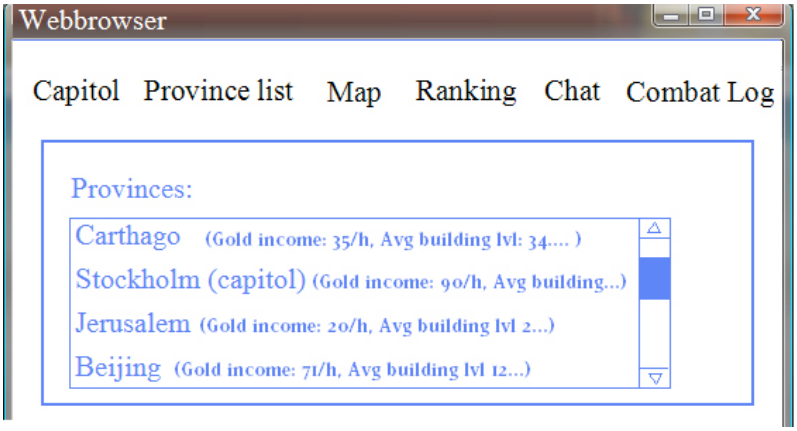


Fig 6.1.2.b

The Map view

The map is a very important part of the game since it represents the overall goal of the game: to capture territory for you own faction.

The map contains of a 100*100 square grid. Each square has 2 properties; it can contain a town, and it belongs to a faction *or* no faction.
When the user selects the "Map" tab in his browser his view (main window) changes to a map. The map is a 10*10 sub grid of the real 100*100 map. This is a rough example:

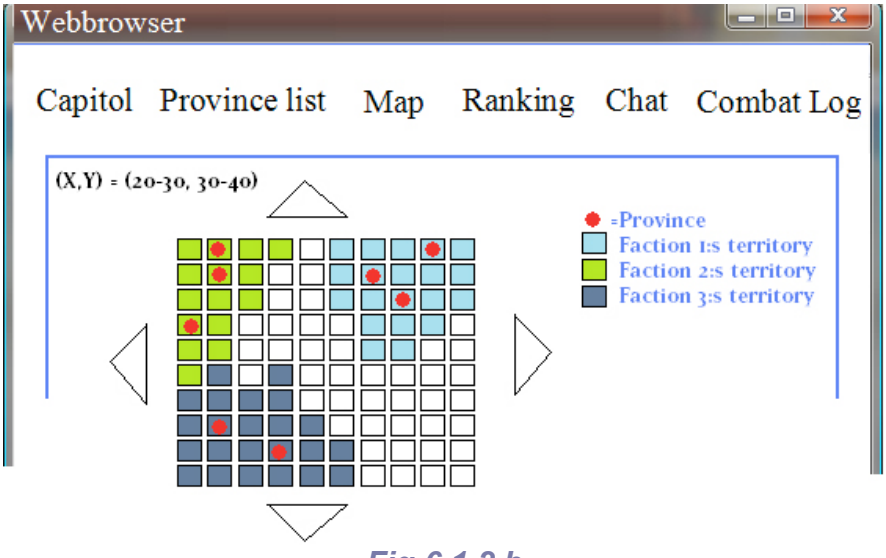


Fig 6.1.2.b

In this view the user can hold his or her mouse pointer over a spot on the map to get information about it. This information includes the owner of a province and the faction that it belongs to. The user can scroll by using the arrows (see the picture). This moves the current view by changing the coordinates of the top left square. The user can also view more details of a square containing a province by clicking on it. These details will include:

- The military strength of the province. Measured as Low, Medium or High. This is calculated relative the overall game standard.

- The average level of all the buildings in the province.
- The amount of resources in the province. . Measured as Low, Medium or High. This is calculated relative the overall game standard.

When the user clicks on a “square”, containing a province, he will also be given the option to send a “private message” to this player.
The map changes when a province is captured by an enemy faction player or when someone builds a new province.

The Ranking view

The ranking is a way to represent how each player is doing compared to other players. Each player will be given a ranking according to his or her in-game process. The ranking includes:

- **Total income from all provinces**: The sum of the income per hour from all the provinces that belongs to that player.
- **Average building lvl:** The average “lvl” of all the buildings that belongs to that player.
- **Military stats:** A measurement of how strong that player’s total army is. (see army for more details)
- **Total amount of provinces.**
- **Overall:** A combined score of all the previous points. Basically a weighted sum of all the above.
- **Ranking:** Shows which player that has the highest overall score by descending order. 1 is highest and 2 is second highest etc.
- **Name:** The name of the player

This score is represented by a scoreboard that is updated every day. The ranking scoreboard can be viewed by the user at any time by clicking on the “Ranking” tab in the in-game environment.
This is a rough example of the scoreboard:

Ranking	Overall	Name	Military	Provinces	Build. lvl	Income
1	67890	Olle	3289	7	45	670/h
2	329	Per	789	2	4	20/h

Table 6.1.2.a

The Combat log view

When the user presses the “Combat log”-tab his main window changes to a mailbox-like environment. This window will contain a list and a message field. When the user clicks on an item in the list it is displayed in the message field. The list contains all “combat logs” from all the combats that the players units have fought as well as other important in-game information. The combat log might look something like:

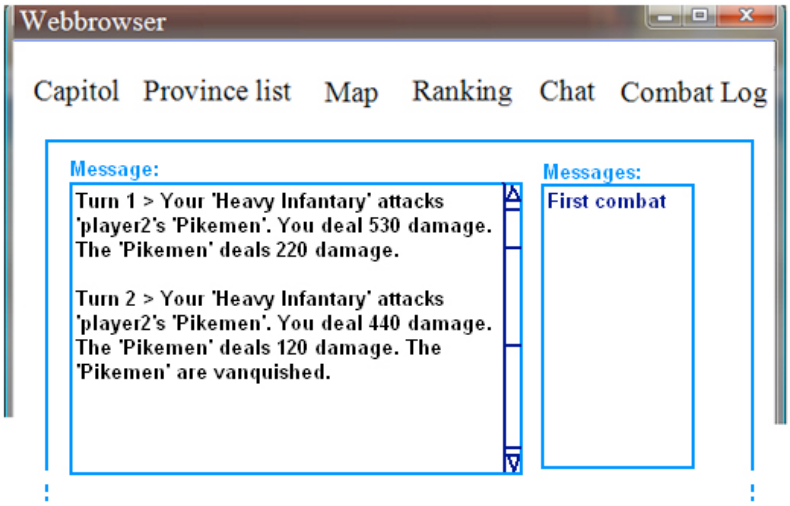


Fig 6.1.2.c

6.1.3 – Build buildings

The Towncentre building

This building is the very core of a province. This building has some important properties such as:

- You can build or upgrade buildings here by selecting one from a menu and clicking on a "build/upgrade" button.
- The only building that you don't have to build. You start with a "lvl 1" "towncentre" in each province.
- You can train units here by selecting the unit you want to train from a menu and clicking a "train unit" button. This option is not available if your town cannot "support" more units or if you don't have enough resources to train them. This subtracts the required resources from your town's total resources.
- You can manage your army here.
- Your gold income is based on the "level" of this building.

6.1.4 – Train units

The Properties of a unit

A unit represents a part of an army. A unit has some very important attributes:

- **Attack**: Modifier when calculating damage during combat (see combat).
- **Defense**: Modifier when calculating damage dealt to your unit by your opponent's unit.
- **HP**: Modifier for damage. If this modifier reaches zero the unit is considered vanquished (see combat).
- **Max HP**: The maximum amount of HP that the unit can achieve.
- **Speed**: How far a unit can travel every hour.
- **Level**: What level the current unit is at.
- **Experience**: How much experience the unit needs to reach the next level.

(*All of these attributes (except speed) only affect the "War"-event which will be explained in detail later on.*) A unit starts at "Level 1" and increases his level by gaining experience. A unit can gain experience by damaging enemy units or by practicing in the "Towncentre" or "Magetower" building. When a unit increases his level some of his attributes increase. These attributes are; Attack, Defense, Max HP, Speed and Level. The maximum level of a unit is 100.
A unit can also have any of the following "traits" that increase each 5th level:

- ***Critical Strike***: Increases the maximum damage of the unit.
- ***Swift Blade***: The unit deals damage to an enemy unit before it has a chance to deal damage to it.
- ***Twin Blades***: The unit deals both "Swift Blade" damage and normal damage.
- ***Berserk***: The damage that is done is calculated by the max-hp.
- ***Trample***: If the unit vanquishes an enemy unit, the remaining damage is dealt to another unit in the enemy army.

There are also 5 "types" of units: Archers, Heavy Infantry, Pikemen and Cavalry. These units have specific bonuses against one another. These bonuses are represented in the picture below.

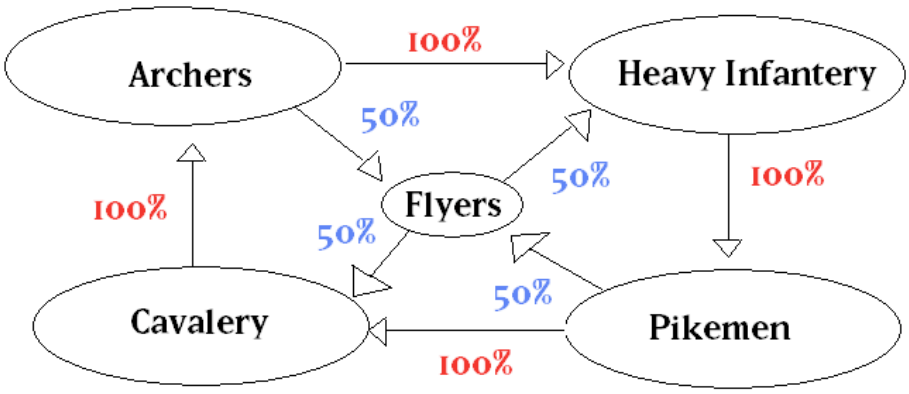


Fig 6.1.4.a

As you can see Cavalry has a 100% damage bonus against Archers, however Archers has a 100% damage bonus against Heavy infantry and so on. Archers and Pikemen seam to have more bonuses then the other unit types, however these units move slower on the map then other units.

6.1.5 – Chat with other players / Send private messages to other players

The Chat view

When the user clicks on the Chat-tab in the in-game environment the main window changes to a new window. This window contains a mailbox and an instant chat window. In this window the user can:

- Send private messages (in-game mail) to another user.

- Read private messages.
- Manage his inbox by deleting or sorting his in-game mail.
- Reply to any in-game mail
- Chat with other players.

The chat contains different channels where users can communicate. A user can change the channel that he is using by selecting one from the "channels list". The user chats by writing a string in the appropriate field (*see Fig 6.1.5a*) and pressing "ENTER". Here is a rough example of how the layout may look like:

Fig 6.1.5.a

Scrolling is enabled if any of the field should be too small to contain all the information. A player can only chat in one channel at the same time.

6.1.6 – Manage armies

Managing armies

A unit can be sent on a mission only if it is a part of an army. Each province can support 2 armies. Each province starts with two empty armies. Units can be moved between armies freely. A unit can be deleted from the army, in which case the user will be asked to confirm this action. If the user presses the "confirm" button then the unit is deleted from the army and also from the game. If the user clicks on the "abort" button the delete-request is discarded. One army is assigned to defend the province against enemy forces until defeated. When the defending army is defeated, or if there is no defending army, the "offensive army" will start defending the province. A defending army cannot be sent to attack another player's province. A player may send units to defend a province that belongs to a player of the same faction. An army has room for 9 units.

Missions

When an army contains at least 1 unit and is positioned inside one of your provinces it can be sent on a mission (against another player's province). Before sent in to war they must be given precisely one of the three missions:

1. **Pillage**: Steal some of the opponent's resources. The user specifies how many turns the army should fight before they try to retreat with some resources. This army will be on the attacking side.
2. **Conquer**: The province is taken from the defending player. This province is considered a new province for the attacking player but some of the buildings will remain at the same level. This army will be on the attacking side.
3. **Defend**: If the province belongs to a player of the same faction the player is given the option to defend that province. If another player sends an army on a "Conquer" mission towards that province this army will help defending it against the attackers. The player specifies how long the army will defend the province (*in hours*). An army on a defend mission will be on the defending side.

The conquer mission / War

Whenever an attacking army reaches a war it gets a timestamp. This timestamp represents his queue in the line to succeed in missions. During a war there will always be two sides: The defending side and the attacking side. The attacking side contains of all attacking armies (can contain of units from different players) and the defending of all defending (can contain units from different players) and also a guardian.

War - The guardian

The guardian is considered a unit during the war. It has all the attributes that a unit has. It can be killed during a war but it will (unless the province is taken) be resurrected after each war. This way the defending province will always have a defending unit. The guardian's strength depends on which mission the attacking player has. The guardian also grows in strength if the defending player has a high "lvl" "Towncentre". If the attacking player has the intention to "pillage" then a relatively weak guardian is summoned. If the mission was to "conquer" the province then a relatively strong guardian is summoned.
(These guardians are made to prevent abuse and harassing of new players)

War – The actual combat

The combat will be turn based. A pillage mission ends after an amount of turns specified by the attacking player. A conquer mission will continue until the attacking army/ies is defeated or until every defending army is defeated including it's guardian.
The army with the lowest timestamp will (if he had chosen the conquer mission and is not vanquished to the last unit) conquer the province.
If many armies have chosen to pillage (and have won the war and have units remaining in the army) the army with the lowest timestamp will pillage first then the one with the second lowest timestamp and so on. If there are no resources to pillage in this province then no resources are awarded by the "pillage"-mission.

When all the missions have been carried out the units return to the provinces that they belong to unless they managed to conquer the province in which case they remain in the province as a defending army.

6.2 Non-Functional requirements specification

As described earlier there are several non-functional requirements for our system. Earlier they were only explained in general but will now be described in a more detailed level and how the system is really affected by them.

6.2.1 Performance requirements

As explained in the user requirements specification the only performance requirement in our system is on the server side of the system.
Of course the time it takes to communicate between user and server depends a lot on the respective connections, but we want the response time from server, after the user performs an action, to be at most around 5 seconds under ideal circumstances.
There are some more specific performance requirements for the system regarding registering, logging in and sending messages.
We want the process of registering to be relatively fast and it shouldn't take more then 5 minutes to fill in the required information. A long registration process can more than often scare away new users.
We also want the login process to be efficient and only in rare cases, (i.e. when the server is almost at it maximum with users), take more than 10 seconds to become fully logged in and able to play.
When using the chat all messages should be received by other users in the chat within 5 seconds. A similar requirement applies to private messages which should be received by the receiving user within 10 minutes of being sent. Of course these last two requirements can be affected by traffic to the server and many users being logged in at one time, but these are only rare cases like with the login process.

6.2.2 Space requirements

Since the game is stored on a server computer there are no requirements for the user regarding how much space is required on his/her hard drive.
The game itself and the information database must be stored on the server computer and we estimate that an amount of 200MB of free hard drive memory should be enough.
Since most modern computers have a lot more memory available this is not really a requirement as to what server computer must be used but more what should be free on it.

6.2.3 Reliability requirements

Since our system must be highly reliable a database will be used to store all of the game information.
To make the system more reliable it must be tested a lot and we must also make sure that 'good' code is written when constructing the program.
If a crash occurs the database will use it's log and backup to restore as much information as possible so the loss and rollback is as small as it can be.

To even further make the system more reliable we have discussed having the database on another computer than the server computer.
If the database does go offline the losses should be minimal; not more then a couple of minutes of information should ever be lost.

6.2.4 Portability requirements

As stated earlier the system must be compatible with most web browsers that can handle modern HTML and JavaScript. We have set no requirement at all that the game will be playable on any other platforms other than an ordinary computer.
The system may be usable on other platforms but it is not supported in any way.

6.2.5 Usability requirements

The system must be easy-to-use and intuitive for the user, and as described earlier we want users with only basic internet knowledge to be able to use our system as well.
Because of this requirement the GUI must be designed in a good way. There shouldn't be too much information and buttons visible at the same time. Objects must be placed in an intuitive way and everything must be visible and clear for the user.

6.2.6 Privacy requirements

Since we don't want the system to show user-information like real life name and e-mail address the system must be designed accordingly. Nothing should be shown to other users other then the username of the user and information about that user's status in the game.
This will be addressed when the system is designed and constructed as it must be fixed in the program.

7. System evolution

7.1 Assumptions on which the system is based

We have not made many assumptions when planning the system. The assumptions that we have made include:

- All modern web browsers support java-script.
- The system can handle 10000 clients (100x100 grid makes the worst-case scenario of 10000 clients being "logged in" at the same time).
- The database can handle up to 50 requests simultaneously without loss of data or peaks in performance. (This is a guess of how many requests that up to 10.000 users would send simultaneously).

7.2 Anticipated changes

Most of the changes in the game will happen because of the demands of the users. Players want the material of the game to be updated regularly in order to be interesting.

7.2.1 Changes due to hardware evolution

We do not expect the hardware evolution to force us to do any upgrades. The dependency of the client's computer performance is minimal since the server does most of the work. It is possible that a server will be able to support much more clients in which case the "map" might be expanded. We will make it possible to expand the map to adjust it to the amount of players and the hardware it runs on.

7.2.2 Changes due to change in user needs

Most of the necessary updates will possibly be because of the user needs. There might be "bugs" in earlier versions of the game that needs correcting and later on the players will demand more game material. We need to keep the game fresh, in the sense that new game material is added regularly, to keep the clients. The game needs to constantly add new challenges and things to learn. The game will be programmed in such a way that game material may be added later.

Changes that may be added:

- **Cross-unit-types**: A unit may be of more than one type and have the benefits and weaknesses of both types. Ex. A unit may be both Cavalry and Archer thus producing a Horse archer. These units may be fast like cavalry, effective against heavy infantry but might be weak against regular cavalry.
  The enabling of cross-unit-types will add new challenges for the players.

- **Increased level cap**: The level cap of 100 might be raised so that players can enjoy the game for a longer period. This will probably attract the hardcore players since they most probably will reach the maximum level with their units.
- **More factions**: More factions could be added to the game. We will make this easy to implement in the future.
- **Hero units**: Units with advanced options when leveling up. A unit that the player can customize in his own way. These units could be highly customizable and have unique traits.
- **New buildings:** Will most probably be added to the game. We will just produce a minimal set of buildings to start with but we plan to expand the number of buildings that you can build. This has to be taken into consideration when coding this game.

We also plan to add advertising banners to the game to get some sort of income from it. This income will hopefully be enough to support its maintenance.

8. Appendices

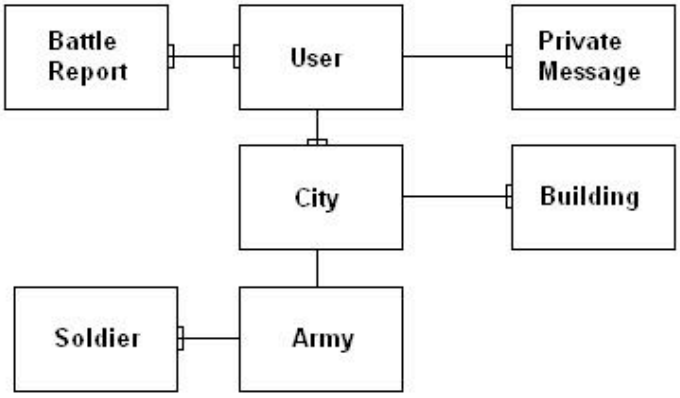
8.1 System requirements

Requirements on the user end:

500MHz Intel Processor or equivalent AMD processor
128 MB Ram
Internet connection, at least 56k modem recommended.
Internet Explorer 5+, Mozilla Firefox or other web browser supporting HTML and recent versions of JavaScript.

8.2 Database design

A preliminary look of the database to be used:



9. Index

Terms and sections not specified in the main index in the beginning of the document.