Software Engineering Project Report

Lithos

97

Group 22
Aravind Achanta
Yonathan Gordon
Naga Kameswari Spoorthi Pendyala
Sai Priya Jyothula
Sumanth Reddy Pandugula
Vakkalanka V S S Dilip Raju

CS 440
at the
University of Illinois at Chicago

Fall 2015

Table of Contents

I Project Description 7

1 Project Overview 7

2 The Purpose of the Project 7

2a The User Business or Background of the Project Effort 7
2b Goals of the Project 7
2c Measurement 7

3 The Scope of the Work 8

3a The Current Situation 8
3b The Context of the Work 8
3c Work Partitioning 9
3d Competing Products 9

4 The Scope of the Product 10

4a Scenario Diagram 10
4b Product Scenario List 10
4c Individual Product Scenarios 11

5 Stakeholders 12

5a The Client 12
5b The Customer 12
5c Hands-On Users of the Product 12
5d Priorities Assigned to Users 12
5e User Participation 13
5f Maintenance Users and Service Technicians 13
5g Other Stakeholders 13

6 Mandated Constraints 13

6a Solution Constraints 13
6b Implementation Environment of the Current System 14
6c Partner or Collaborative Applications 14
6d Off-the-Shelf Software 14
6e Anticipated Workplace Environment 14
6f Schedule Constraints 14
6g Budget Constraints 14

7 Naming Conventions and Definitions 14

7a Definitions of Key Terms 14
7b UML and Other Notation Used in This Document 15
7c Data Dictionary for Any Included Models 15

8 Relevant Facts and Assumptions 15
- 8a Facts 15
- 8b Assumptions 16

II Requirements 17

9 Product Use Cases 17
- 9a Use Case Diagrams 17
- 9b Product Use Case List 17
- 9c Individual Product Use Cases 18

10 Functional Requirements 30

11 Data Requirements 32

12 Performance Requirements 33
- 12a Speed and Latency Requirements 33
- 12b Precision or Accuracy Requirements 34
- 12c Capacity Requirements 34

13 Dependability Requirements 34
- 13a Reliability Requirements 34
- 13b Availability Requirements 34
- 13c Robustness or Fault-Tolerance Requirements 34
- 13d Safety-Critical Requirements 35

14 Maintainability and Supportability Requirements 35
- 14a Maintenance Requirements 35
- 14b Supportability Requirements 35
- 14c Adaptability Requirements 35
- 14d Scalability or Extensibility Requirements 36
- 14e Longevity Requirements 36

15 Security Requirements 36
- 15a Access Requirements 36
- 15b Integrity Requirements 36
- 15c Privacy Requirements 37
- 15d Audit Requirements 37
- 15e Immunity Requirements 37

16 Usability and Humanity Requirements 37
- 16a Ease of Use Requirements 37
- 16b Personalization and Internationalization Requirements 38
- 16c Learning Requirements 38
- 16d Understandability and Politeness Requirements 38
- 16e Accessibility Requirements 39

16f User Documentation Requirements 39
16g Training Requirements 40

17 Look and Feel Requirements 40

17a Appearance Requirements 40
17b Style Requirements 40

18 Operational and Environmental Requirements 41

18a Expected Physical Environment 41
18b Requirements for Interfacing with Adjacent Systems 41
18c Productization Requirements 41
18d Release Requirements 41

19 Cultural and Political Requirements 41

19a Cultural Requirements 41
19b Political Requirements 41

20 Legal Requirements 42

20a Compliance Requirements 42
20b Standards Requirements 42

III Design 43

21 System Design 43

21a Design goals 43

22 Current Software Architecture 43

23 Proposed Software Architecture 43

23a Overview 43
23b Class Diagrams 44
23c Dynamic Model 44
23d Subsystem Decomposition 47
23e Hardware/software Mapping 48
23f Data Dictionary 49
23g Persistent Data management 50
23h Access control and security 50
23i Global software control 50
23j Boundary conditions 51

24 Subsystem services 51

25 User Interface 52

26 Object Design 52

26a Object Design trade-offs 52

26b Interface Documentation guidelines 52
26c Packages 53
26d Class Interfaces 53

IV Test Plans 54

27 Features to be tested / not to be tested 54

28 Pass/Fail Criteria 54

29 Approach 54

30 Suspension and resumption 54

31 Testing materials (hardware / software requirements) 55

32 Test cases 55

33 Testing schedule 59

V Project Issues 59

34 Open Issues 59

35 Off-the-Shelf Solutions 60

35a Ready-Made Products 60
35b Reusable Components 60
35c Products That Can Be Copied 60

36 New Problems 61

36a Effects on the Current Environment 61
36b Effects on the Installed Systems 61
36c Potential User Problems 61
36d Limitations in the Anticipated Implementation Environment That May Inhibit the New Product 61
36e Follow-Up Problems 61

37 Tasks 62

37a Project Planning 62
37b Planning of the Development Phases 62

38 Migration to the New Product 62

38a Requirements for Migration to the New Product 62
38b Data That Has to Be Modified or Translated for the New System 62

39 Risks 62

40 Costs 63

41 Waiting Room ... 63

42 Ideas for Solutions ... 63

43 Project Retrospective ... 64

VI Glossary ... 64

VII References / Bibliography ... 65

I Project Description

1 Project Overview

Lithos is an interactive multiplayer strategy game in which the user plays the role of a tribal leader who tries to overcome various obstacles in Prehistoric Age. The game aims to add educational value to gaming environment by helping the player learn Prehistory as he/she progresses through various levels of the game.

2 The Purpose of the Project

2a The User Business or Background of the Project Effort

What if a game can be used as a means of recreation as well as instruction? Lithos is the answer to that question – it is intended to teach the user about the events of Prehistoric Age in an engaging gaming environment, essentially providing the user with "edutainment".

In 21st century, less number of individuals is aware of the foundation of human evolution and history. Many games that are designed to educate the user in history concentrate on the amusement aspect of the game and compromise on historical accuracy by showing some anachronisms. The game Lithos focuses on being informative and accurate as well as entertaining. The key idea of Lithos is to teach individuals about Prehistory and preserve the knowledge of development of humanity for future generations.

2b Goals of the Project

The goal of the project is to educate high school students and other interested individuals about the Prehistoric Age in human evolution.

The motivation behind Lithos is to convey Prehistoric chronology of events to youngsters and help them to understand how early humans handled situations such as threats, hunger and hardships.

Examples

The user must be able to know the facts and features of the particular era after completing the corresponding level in chronological order. For instance, if the user completes the level corresponding to Paleolithic age, then he/she must know about thedevelopments that occured in that period of time like the discovery of fire, wheel and sartorial realization.

2c Measurement

In schools, the success of the product can be measured by checking the performance of students in the quizzes within the game and the actual Prehistory quizzes conducted in their respective classes. If the performance of the students improves through this game, then that particular school will recommend it to the rest of the schools which helps in growth of potential customer base for the product.

In case of the individuals that play the game due to their interest in knowing about Prehistory, the success can be measured based on the number of hours of game play, the number of shares of the quiz scores and game progress to their friends. The average customer rating can also be used to measure the success of our product.

3 The Scope of the Work

3a The Current Situation

Existing prehistoric games like ‘Flintstones’,’ PreHistoric Tribes’ mainly focus on engaging the user rather than educating the user about the facts of prehistory. These games also tend to distort the facts for the sake of making the game more interesting. Also, they concentrate on only one period in the Prehistoric Age, so user will not be able gain an in-depth knowledge of how the initial tribes made it to the established civilizations.

Our game focuses on engaging the user as well as educating him/her about the different strategies used by the early tribes to survive the unfavorable conditions. Lithos also takes into account the exact chronology of events, discoveries and behavioral advancements in early tribes.

3b The Context of the Work

<u>Examples</u>

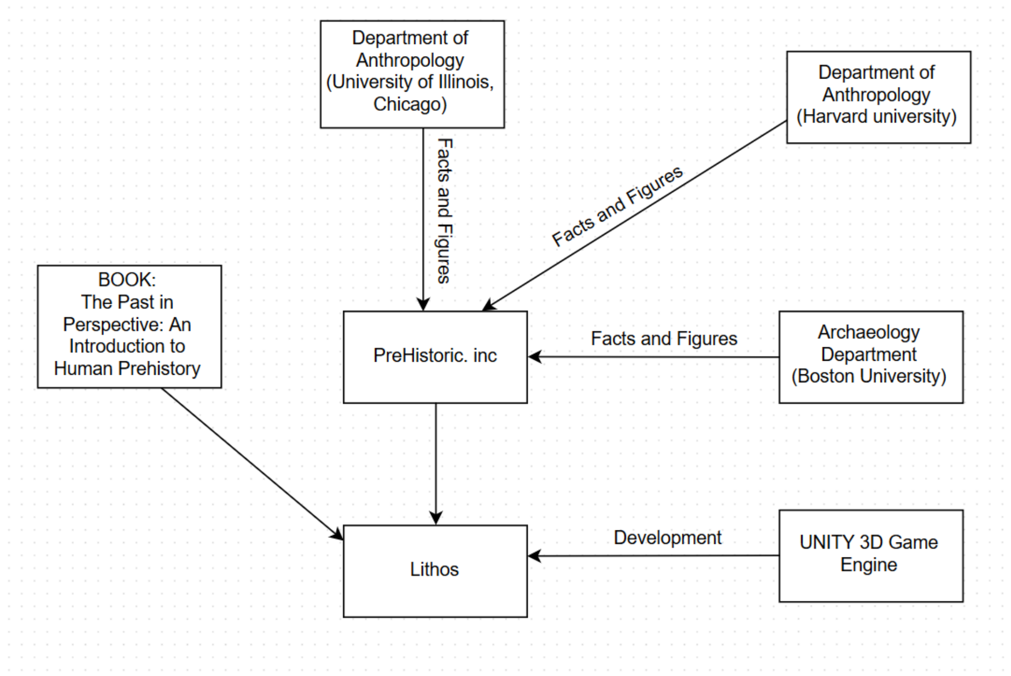

Department of Anthropology (University of Illinois, Chicago)
Department of Anthropology (Harvard university)
Facts and Figures
Facts and Figures
BOOK: The Past in Perspective: An Introduction to Human Prehistory
PreHistoric. inc
Facts and Figures
Archaeology Department (Boston University)
Lithos
Development
UNITY 3D Game Engine

Figure 1 – Context

3c Work Partitioning

EVENT NAME	INPUT AND OUTPUT	SUMMARY
Game Loading	Input - .exe file Output - Game Home Screen	Loads the game.
Display Main Menu	Input- User choice from the set of options available Output-Redirection of the game accordingly	The Main menu is shown at the startup which enables the user to choose from the different set of game options, like controls.
Level Ups	Input- Successful completion of the previous level Output- Next Level Screen	Goes to the next level when the previous level goals are achieved.
Show Clues	Input- The user's current difficulty Output- Help tips to take the player further in the game	If the user gets stuck at some point or the player takes more time to realize his next move, the hints start popping up.
Show Notifications	Input- The user's current status in the gameplay Output- The appropriate notification	Whenever there is a need to inform the user of his progress in the game/ if he is running on low resources, notifications are shown.

Table 1 - Work Partitioning

3d Competing Products

Games like "Flintstones" and "PreHistoric Tribes" belong to the same genre of games, but they compromise totally on historical facts, there by degrading the educational value. For instance, in Flintstones early humans co-exist with dinosaurs – this is a historically inaccurate event and there by misguides the user. Lithos solves this by providing the user with accurate prehistoric information which is collected from trusted sources (PreHistoric Inc.).

4 The Scope of the Product

Lithos is a strategy game that will be developed for a PC platform, but it can be potentially ported to a gaming console or even a mobile device that would meet the hardware requirements. The hardware that would interface with the game will be a mouse, a keyboard and/or a controller. The product primarily targets gamers, as well as individuals who are interested in the historical content provided in the game and are looking to explore gaming. The main goal is to make the product available through a gaming platform/website such as "Steam", since it is the best way for the game to gain popularity and maximize revenue.

4a Scenario Diagram

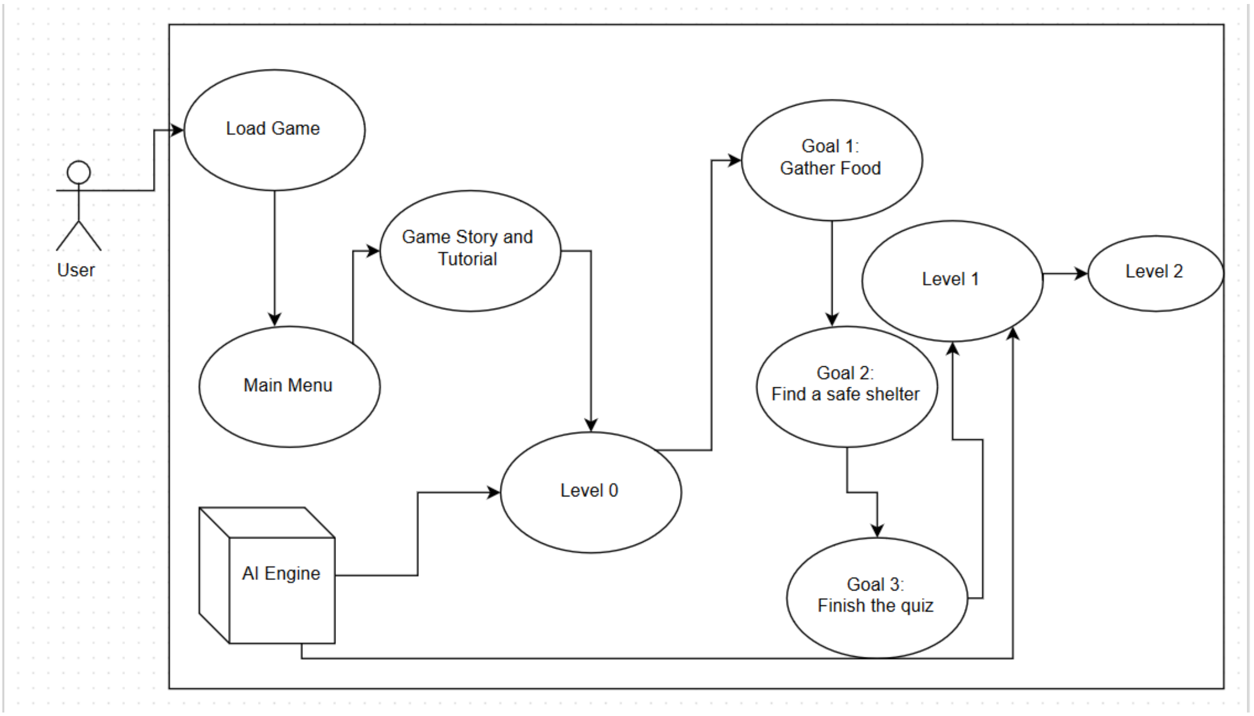


Figure 2 - Scenario Diagram

4b Product Scenario List

1. Launch Game
2. Single Player
3. Multi Player
4. Manage Resources

5. Explore Levels
6. Level-up quizzes

4c Individual Product Scenarios

1. **Launch game:** The game is launched by double clicking on the executable file. If the launch takes time, then a progress bar pops up, which assures the user that progress is being made. The game starts off with a main menu which has a captivating background and multiple options like Choice of player, New Game, Load Pre Existing game, Settings, Help and Exit. For example, if the user clicks on Load Pre Existing game, then it will allow the user to load the pre saved file. Settings has an option to control screen size, volume, user profile, mouse settings, key board controls, camera settings and other input device settings. Exit is a typical log out set up when the user has to confirm about saving the current game and exiting.

2. **Single Player:** If the application is used in a single player mode, the user will be playing against an AI generated characters that will try to beat the user through his quest to create civilization. The opponent capabilities can be adjusted to different levels depending on the needs of the user. As the user progresses through the ages, the AI generated characters become more skillful and crafty as well making it harder for the user to achieve his goals.

3. **Multi Player:** Lithos will also allow for a multi-player mode by allowing multiple players to use the application by connecting to the AI engine through internet. The AI engine will look for players online who are at the same skill level and it gives recommendations to the player to conquer or form alliance with other player's tribe.

4. **Manage Resources:** User will have certain resources for his disposal, such as food, water, and other basic ones needed to start. The game progresses based on the amount of resources a user maintains and this is monitored through a set of meters.

5. **Explore Levels:** The game progresses to the next level after completing a set of challenges. Additional challenges that are historically accurate are unlocked with each new level. For example fire becomes available to the user immediately after completing the Level 0 (Lower Paleolithic Age) and fire has to be used to complete the challenges in level 1.

6. **Level-up Quizzes:** After achieving the challenges specific to a level, the user is tested on historical knowledge gained in that level through a series of questions. Successful completion of the quiz is mandatory for progressing to the next level.

5 Stakeholders

5a The Client

The client is LITHOS.INC, which is a company that is focused on providing path breaking practical experience to people who want to learn history, especially concentrating on prehistoric events through edutainment.

5b The Customer

The intended customers of this product are all schools that teach history to students. It will also be available to enthusiastic people who want to learn about prehistoric events. To be particular, this product will improve the knowledge of students considerably as the product teaches in a both engaging and informative way.

5c Hands-On Users of the Product

The end users of the product are high school students. Students will play the game Lithos and gain knowledge in Prehistory. Other hands-on users would be the teachers who test the knowledge obtained by the student and constantly monitor them.

Students: Students who play Lithos will gain knowledge regarding the prehistoric life of humans and the events in Prehistory in a chronological order through a level by level approach. Students will learn a lot of information about how early humans used to think in case of need, how they thrived in harsh environment and overcame the hurdles.

Teachers: The game Lithos displays a quiz at the end of each level and the players are required to answer the questions satisfactorily to move on to the next level. It is the role of the teachers to monitor quiz performances of the students and gain an overview of how much students have learnt from the game.

5d Priorities Assigned to Users

- **Key users:** Students (80% of the users)

This game is used by students to gain knowledge in Prehistory – this should be done in entertaining yet informative manner.

- **Secondary users:** Teachers (10% of the users)

Teachers use this application to get an overall view of students' performance in game quizzes and in class.

- **Other users:** Sales and Admin (10% of users)

Individuals who seek to learn more about Prehistory will use this game for this purpose. Representatives and administrators use this product for deploying and maintaining the product sales.

5e User Participation

There will be very less participation of the primary user as he/she gets educated from the software, but there is a very good scope for the secondary users to assist product developers by giving few insights on the improvement of the product.

5f Maintenance Users and Service Technicians

Very minimal software management is required. Lithos Inc. will offer product maintenance for a small fee on any errands that may arise in the future.

5g Other Stakeholders

- **Statisticians, Economists**: These stakeholders are essential for growth of the product as they analyze the sales of the product and provide an insight into the economic standards of the company and product.

- **Lawyers:** For any infringements and copyrights regarding the product, these stakeholders take care of the security issues related to the product.

6 Mandated Constraints

6a Solution Constraints

Constraint 1:

Description: The game should use accurate information about prehistoric timeline.

Rationale: The game is meant to teach history, so any false information concerning historical facts embedded into the game will not result in the product being what it was meant to be.

Fit Criterion: The information provided by the Pre Historic Inc. is compared with data from Wikipedia and like sources for a match of 99%.

Constraint 2

Description: The game should be highly engaging and easily comprehensible.

Rationale: This game's intent is to teach prehistoric facts and as most of the users are high school students, it is a must for the game to be both educating and engaging.

Fit Criterion: The application provides the user with the option of rating the game (on a scale of 0 to 5). If the average rating is equal to or greater than 4, the game is considered highly engaging and comprehensible.

6b Implementation Environment of the Current System

The game runs on Windows, Mac OS X and Linux. Even though the main market these days is Windows, the company plans to increase the customers in future who use both Mac and Linux for profits and popularity. It also connects to an online game server for multiplayer mode.

6c Partner or Collaborative Applications

This game does not collaborate with any other applications at the initial stage. But as the levels progress, it enters a multi-player mode which connects the player to the online game server (player match making app). The user can then compete against other players or collaborate with them as needed.

6d Off-the-Shelf Software

This game doesn't use any off-the-shelf software.

6e Anticipated Workplace Environment

As this game has some educational value attached to it, it is mostly played by students in schools or individuals that are interested in learning Prehistory. The system that runs the game allows the player to use a headset in order to avoid inconvenience to his immediate surroundings. The user can even play it on his personal computer according to his preference.

6f Schedule Constraints

The game is supposed to be developed by a team within 3 years of receiving the final report. As the target audience is exposed to the ever changing scenario of graphical world, it is essential that the game must be deployed in the timeframe defined so that it competes with the present day graphics.

6g Budget Constraints

As this game requires a lot of resources, the budget estimation is not fixed and will change depending upon the scenarios. This is undetermined as of yet.

7 Naming Conventions and Definitions

7a Definitions of Key Terms

Stone Age: This is the first stage in the prehistoric chronology of events.

Paleolithic Age: This is the first stage in the Stone Age that is further divided into 3 stages mentioned below.

Lower Paleolithic: This is the first stage in Paleolithic Age where early humans focused on basic needs like food and shelter, i.e., survival.

Middle Paleolithic: This is the second stage in Paleolithic Age where early humans focused on needs like Clothing and this stage also marks the discovery of fire.

Upper Paleolithic: This is the third stage in Paleolithic Age where early humans focused on strategic usage of fire for cooking, dealing with threatening animals and providing warmth.

Mesolithic Age: This is the second stage in the Stone Age where early humans learnt skills like stone carving and land technology.

Neolithic Age: This is the third stage in the Stone Age where early humans learnt techniques of agriculture and taming animals.

Bronze Age: This the second stage in the prehistoric chronology of events where skills like trading using barter system, invention of Potter's wheel and navigation got developed.

Iron Age: This is the third stage in the prehistoric chronology of events where the people have started using Iron to make weapons.

7b UML and Other Notation Used in This Document

This document generally follows the Version 2.0 OMG UML standard, as described by Fowler in "UML Distilled", Third Edition, Boston: Pearson Education, 2004.

7c Data Dictionary for Any Included Models

Not Applicable at this point of time.

8 Relevant Facts and Assumptions

8a Facts

- Stone Age is the earliest known age.
- Lower Paleolithic age dealt only with humans searching for food and shelter.
- Middle Paleolithic age is where humans have little knowledge about clothing.
- Fire was discovered in the middle Paleolithic age due to natural lightning.
- The first metal, Copper was known to humans in early Bronze Age.
- Humans learnt about alloys in middle Bronze Age.
- People started using Iron for weaponry in the Iron Age.

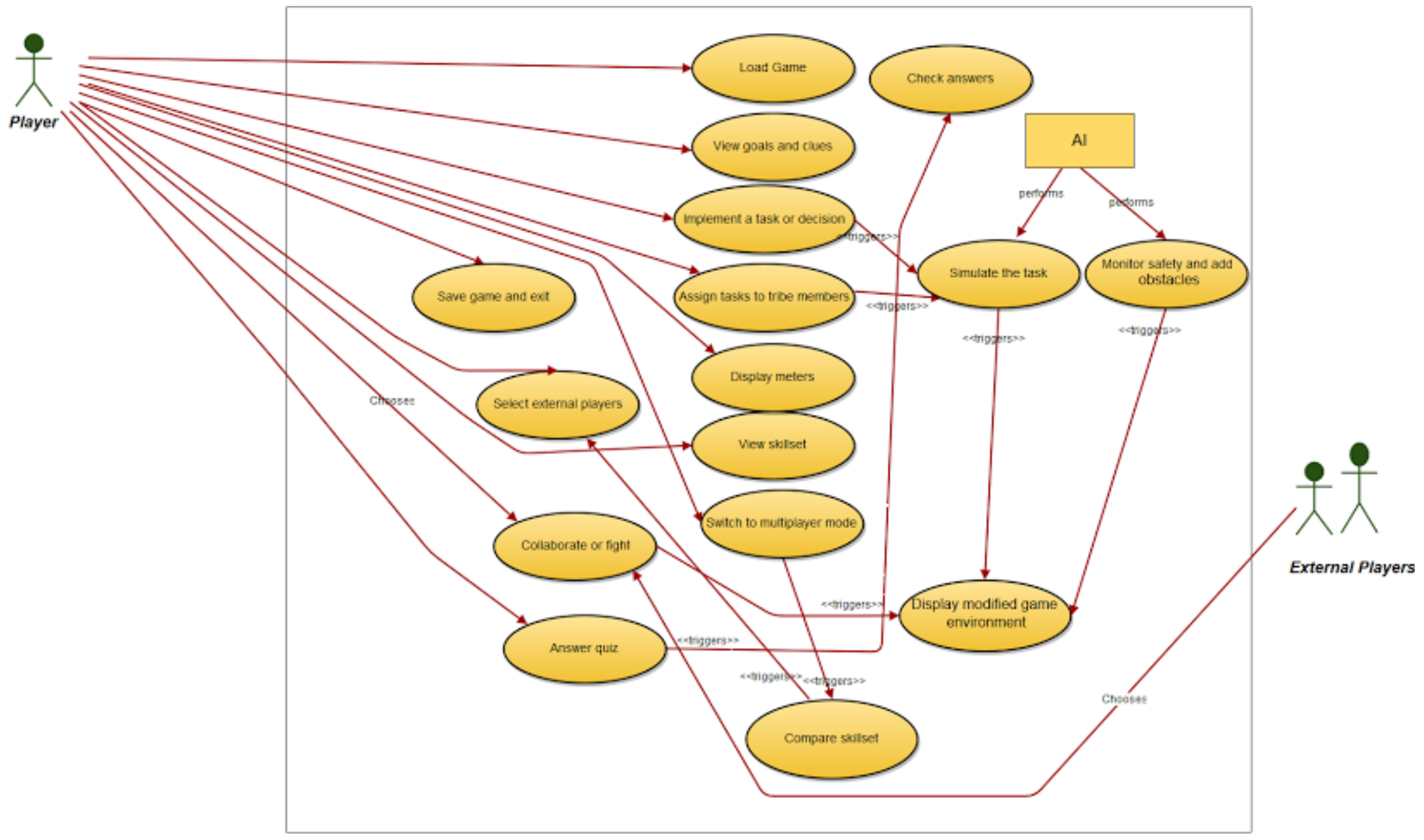
8b Assumptions

- The game targets the PC platform since it has become the most popular platform for gaming.
- Internet connection is required on the users' end to play in a multi-player mode.
- PC hardware is getting cheaper allowing for more people to purchase their own gaming machines and high end PCs.
- The game is historically accurate and presents the facts in the correct chronological order
- The game should attract both dedicated gamers and casual gamers and it should gear towards all age groups.
- The game will be available for all Desktop platforms (Windows, Mac, Linux)
- The game will be programmed using the game development engine "UNITY 3D".
- User data will not be released, i.e., confidential and always be kept for internal purposes.
- A basic prototype showcasing the look and feel of the game should be released within the first three months.

II Requirements

9 Product Use Cases

9a Use Case Diagrams



9b Product Use Case List

i. LoadGame

ii. ViewGoalsAndClues

iii. ImplementATask

iv. AssignTasksToTribeMembers

v. SimulateTheTask

vi. DisplayMeters

vii. ViewSkillset

viii. SwitchToMultiplayerMode

ix. CompareSkillset

x. SelectExternalPlayers

xi. CollaborateOrFight

xii. DisplayModifiedGameEnvironment

xiii. AnswerQuiz

xiv. MonitorSafetyAndAddObstacles

xv. CheckAnswers

xvi. SaveGameAndExit

9c Individual Product Use Cases

Use case ID: 1 Name: LoadGame pre-conditions: The application is opened, the user is logged into the system and at the main menu. post-conditions: The initial game environment is displayed. Initiated by: Player Triggering Event: The player selects the 'Load Game' option. Additional Actors: Not applicable
Sequence of Events: 1. The player selects the 'Load Game' option. 2. The system prompts the player to choose one of the two options – either to start a new game or to load a pre-existing game. 3. The player chooses one of the two options according to his/her preference. 4. Based on the player's preference, the system displays a new game or checks the game log of the user and displays the pre-existing game.
Alternatives: The player aborts the currently playing game and chooses to load a new or pre-existing game.

Exceptions: 1. The previous game of the player is saved incorrectly.

2. The player doesn't have a saved game.

Use case ID: 2 Name: ViewGoalsAndClues

pre-conditions: The player is playing in a particular level.

post-conditions: The system displays the goals of the particular level and any clues that may help the player to progress through the particular level.

Initiated by: Player

Triggering Event: The player selects the 'View Goals and Clues' option.

Additional Actors: Not applicable

Sequence of Events:

1. The player selects the 'View Goals and Clues' option.
    2. The system displays the goals of the current level and provides the player with relevant clues according to the challenges of the level.
    3. The system provides a means to close the displayed information.
4. The player chooses to close the information.
    5. The system returns to the game environment.

Alternatives: 1. The system displays the goals at the beginning of each level.

2. The player is not able to complete the goals within the specified time and the system reminds the player of the goals to be met or provides any relevant clues.

Exceptions: The player tries to view goals at the quiz section of the game.

Use case ID: 3 Name: ImplementATask

pre-conditions: The player is playing in a particular level and encounters a challenge.

post-conditions: The system displays the game environment after implementing the decision made by the player.

Initiated by: Player

Triggering Event: The player implements a task by selecting the corresponding option.

Additional Actors: Not applicable

Sequence of Events:

1. The player implements a task to overcome the challenge by selecting the corresponding option.

   2. The system displays the task being implemented.
   3. The system displays the game environment after the task is implemented.
   4. The system notifies the player of the next challenge.

Alternatives: The player may re-implement the task if he/she is not satisfied with the score of the task.

Exceptions: 1. The previous task is incomplete.

2. The implemented task is irrelevant to the challenge.

Use case ID: 4 Name: AssignTasksToTribeMembers

pre-conditions: The player is playing in a particular level and lacks certain resources.

post-conditions: The tribe members work on assigned tasks and result in a modified game conditions.

Initiated by: Player

Triggering Event: The player selects the ‘Assign tasks’ option.

Additional Actors: Not applicable

Sequence of Events:

1. The player selects the 'Assign tasks' option.

Loop:

2. The system displays a table with the following columns – 'tribe member', 'existing skillset', 'assigned tasks', 'potential tasks that can be assigned to the tribe member'.
3. The system provides a means for the player to choose a tribe member.

4. The player selects a tribe member.

5. The system provides a means for the player to assign the tasks to the tribe member.

6. The player selects and assigns tasks to the tribe member.

7. The system updates the table based on assigned tasks and goes to Loop until the player assigns the tasks for all tribe members.

Loop Ends

8. The system displays the overall changes in the skillset of the tribe and modified game environment (such as increase in resources like food or improved shelter).

Alternatives: The player might want to improve the skillset of tribe members by assigning tasks.

Exceptions: 1. The tribe member doesn't have any potential tasks.

2. The player cannot possibly assign tasks while more pressing challenges are under way.

Use case ID: 5 Name: SimulateTheTask

pre-conditions: The player is at a particular level and encounters a challenge.

post-conditions: The simulation of the tasks is performed and the simulation reports are generated by AI.

Initiated by: AI (internal actor because AI is a part of the system)

Triggering Event: The player implements a task or assigns tasks to tribe members.

Additional Actors: Player

Sequence of Events:

1. The player implements a task or assigns tasks to tribe members (trigger).

    2. The AI performs the simulation of tasks and makes decisions about how these tasks solve the game challenges or affect the game environment.
    3. The AI creates the reports of the simulation for the use of the system. These reports include the details of the challenges (or goals of the particular level) that are solved by the implemented tasks.

Alternatives: In multiplayer mode, the external players may implement tasks or assign tasks to their tribe members.

Exceptions: 1. The task that is assigned to a tribe member is not under his/her capabilities.

2. The assigned task doesn't result in a change of the game environment.

Use case ID: 6 Name: DisplayMeters

pre-conditions: The player started the gameplay.

post-conditions: The system displays various meters (such as food and safety) showing accurate values.

Initiated by: Player

Triggering Event: The player selects 'Display Meters' option.

Additional Actors: Not Applicable

Sequence of Events:

1. The player selects 'Display Meters' option.

    2. The system checks the current status of the game and quantifies the status quo on a scale of 0 to 100.
    3. The system displays various meters showing the status of food, resources and safety.

Alternatives: The meters are displayed when the value of one of the meters is under the critical value (very less).

Exceptions: The calculated value of the meter is above 100.

Use case ID: 7 Name: ViewSkillset

pre-conditions: The game environment contains at least one tribe member.

post-conditions: The system displays the current skillset of the selected tribe member.

Initiated by: Player

Triggering Event: The player selects ‘View Skills’ option.

Additional Actors: Not Applicable

Sequence of Events:

1. The player selects ‘View Skills’ option to know more information about the particular tribe member.

    2. The system displays the abilities and skillset of the selected tribe member.
    3. The system provides a means to close the displayed information.

4. The player chooses to close the information.

    5. The system returns to the game environment.

Alternatives: The player may view skills of any tribe member whenever that tribe member appears.

Exceptions: 1. The information about the tribe member is unavailable.

2. The tribe member is absent in the current level.

Use case ID: 8 Name: SwitchToMultiPlayerMode

pre-conditions: The player is in single player mode and has access to a network of external players.

post-conditions: The game switches to multiplayer mode.

Initiated by: Player

Triggering Event: The player selects ‘Multiplayer Mode’ option.

Additional Actors: Not Applicable

Sequence of Events:

1. The player selects ‘Multiplayer Mode’ option.

    2. The system searches for available external players within the network.
    3. If external players are available, the system changes the game to multiplayer mode.

Alternatives: The external player may send an invite for the player to switch to multiplayer mode.

Exceptions: The connectivity to the network of external players is disturbed.

Use case ID: 9 Name: CompareSkillset

pre-conditions: The player is in multiplayer mode.

post-conditions: The system arrives at a list of compatible external players.

Initiated by: Time

Triggering Event: The player switches the game to multiplayer mode.

Additional Actors: Player

Sequence of Events:

1. The player switches the game to multiplayer mode (trigger).

    2. The system compares the skillset of the player with the external players within the network.
    3. The system assembles a list of external players which have compatible skillsets with the player.
    4. The system refreshes the list every 30 minutes by going back to Step 2.

Alternatives: Not Applicable

Exceptions: The skillset of the player or one of the other external players is unavailable.

Use case ID: 10 Name: SelectExternalPlayers pre-conditions: The player is in multiplayer mode and the system has the list of compatible external players.

post-conditions: The system sends invites from the player to the selected external players to join the multiplayer game.

Initiated by: Player

Triggering Event: The player selects the ‘Select External Players’ option.

Supporting use cases: CompareSkillset (Use Case ID: 9)

Additional Actors: Not Applicable

Sequence of Events:

1. The player selects the ‘Select External Players’ option.
    2. The system displays the list of compatible external players and provides the player with the option to know more about each of them.
3. The player selects the option to know more about a particular external player.
    4. The system displays the detailed information (such as the highest played level, skillset, number of tribe members, scores) of the particular external player.
    5. The system provides a means for the player to send an invite to the external player to join the multiplayer game.
6. The player chooses to send an invite to the external player.
    7. The system notifies the external player of the invite.
    8. The system provides the player with the option to go back to the list again (Step 2).

Alternatives: One of the external players sends an invite for the player to join the multiplayer game.

Exceptions: 1. The external player does not have comparable skillset.

2. The communication system that sends the invite is down.

3. The detailed information about one of the other external players is unavailable.

Use case ID: 11	Name: CollaborateOrFight

pre-conditions: The player sends invites to the compatible external players.

post-conditions: The system establishes the relationships between the player and other external players.

Initiated by: Player

Triggering Event: The player selects the ‘Collaborate or Fight’ option.

Additional Actors: External players

Sequence of Events:

1. The player selects the ‘Collaborate or Fight’ option.
    2. The system checks the responses (from the external players) for the invites sent by the player.
    3. The system displays the list of external players who accepted the invitation from the player. The external players who got their invitation accepted by the player are also included in the list.
    4. The system provides a means for the player to choose between the two options – either collaborate or fight with the particular external player.
5. The player chooses one of the two options according to his/her preference.
    6. The system checks whether the response of the player is the same as the one given by the external player.
    7. If the responses are same, the system modifies the game environment in the multiplayer mode.
    8. If the responses are different, the system prompts the player to pick another external player or to change the response.

Alternatives: The player may choose to change his/her preference from collaborate to fight or viceversa at the beginning of the next level.

Exceptions: The response of the external player is unavailable at the moment.

Use case ID: 12 Name: DisplayModifiedGameEnvironment

pre-conditions: The system simulates the tasks implemented by the player.

post-conditions: The system displays the modified game environment or end-of-level quiz.

Initiated by: Time

Triggering Event: The player makes changes to the game environment by implementing or assigning tasks.

Additional Actors: Player

Sequence of Events:

1. The player makes changes to the game environment by implementing or assigning tasks (trigger).
    2. The system checks the simulation reports generated by the AI as soon as they are available.
    3. The system computes the progress of the level (such as the number of goals met) based on the simulation reports.
    4. If all goals of the level are met (level complete), the system displays the end-of-level quiz.
    5. If some goals of the level are remaining, the system displays modified game environment accordingly.

Alternatives: The AI may add new obstacles to the game in periods of safety to maintain the competitiveness of the game which result in the modification of game environment.

Exceptions: 1. The simulation reports are unavailable.

2. The implemented tasks are not relevant to the level.

Use case ID: 13 Name: AnswerQuiz

pre-conditions: The player meets all the specified goals of the particular level.

post-conditions: The system provides a means for the player to answer the end-of-level quiz and stores the player responses.

Initiated by: Player

Triggering Event: The player selects the ‘Answer Quiz’ option.

Additional Actors: Not Applicable

Sequence of Events:

1. The player selects the ‘Answer Quiz’ option.

    2. The system confirms whether the player has completed all the goals of the particular level.
    3. If all the goals are met, the system displays the end-of-level quiz questions that focus on the events of the prehistoric age corresponding to the particular level.
    4. The system provides a means for the player to answer the quiz.

5. The player responds to the quiz questions based on his/her ability.

    6. The system stores the responses of the player.

Alternatives: The player may choose to answer the end-of-level quiz of any of the previously played levels.

Exceptions: The quiz question is not relevant to the particular level.

Use case ID: 14 Name: MonitorSafetyAndAddObstacles

pre-conditions: The player started the gameplay.

post-conditions: The difficulty of the game is re-adjusted by the system.

Initiated by: AI (internal actor because AI is a part of the system)

Triggering Event: Time

Additional Actors: Not Applicable

Sequence of Events:

1. The system continuously checks for any changes in the game environment.

    2. If there is a change in the game environment, the system checks the status of safety meter.
    3. If the value of safety is greater than 95 for more than 2 minutes, the AI adds relevant obstacles to maintain the competitiveness of the game.

Alternatives: Not Applicable

Exceptions: The challenges are complicated enough for the player, making the addition of new obstacles redundant.

Use case ID: 15 Name: CheckAnswers

pre-conditions: The player responses for the end-of-level quiz are stored by the system.

post-conditions: The player is progressed to the next level or repeats the current level.

Initiated by: System

Triggering Event: The player answers the end-of-level quiz.

Additional Actors: Player

Sequence of Events:

1. The player answers the end-of-level quiz (trigger).

2. The system compares the responses of the player with the actual answers.
3. The system computes the score based on the matched answers.
4. If the score is above a satisfactory level (90 on a scale of 100) the system progresses the player to the next level.
5. If the score is not satisfactory, the system displays a message informing the user of the low score and restarts the current level.

Alternatives: Not Applicable

Exceptions: 1. The actual answers cannot be retrieved from the database.

2. The stored responses of the player cannot be retrieved.

Use case ID: 16 Name: SaveGameAndExit

pre-conditions: The player starts the gameplay.

post-conditions: The game is saved to the player's game log and the game is closed.

Initiated by: Player

Triggering Event: The player selects the 'Save and Exit' option.

Additional Actors: Not applicable

Sequence of Events:

1. The player selects the ‘Save and Exit’ option.
    2. The system modifies the player’s game log by saving the most recent gaming activity (the system uses the simulation reports) to the database.
    3. The system provides a means for the player to confirm the exit or return to previous screen.
4. The player chooses an option based on his/her preference.
    5. Based on the player’s preference, the system either exits or displays the previous screen.

Alternatives: The player may choose to directly exit the game.

Exceptions: 1. The player doesn’t have previous game log.

2. The player is in the middle of the timed quiz and cannot exit the game.

10 Functional Requirements

The functional requirements are divided into six different sections as specified below. The fit criterion for these requirements are discussed in the Test Cases section of this document.

A. **Load and Save Game:**
The system must provide a means for the player to choose one of the two options at the main menu – either to start a new game or to load a pre-existing game. Based on the player’s preference, the system must display a new game or display the pre-existing game by checking the game log of the user.
The system must modify the player’s game log by saving the most recent gaming activity to the database. The system must provide a means for the player to confirm the exit or return to the previous screen. Based on the player’s preference, the system must exit or return to the previous screen.

B. **Information Screens:**
The system must display a screen with the goals of the current level and provide the player with relevant clues according to the challenges of the level when the player chooses to view goals and clues.

The system must be able to quantify the status quo at the particular point in a level and display the values with the use of various meters (food, resources and safety). The system must display the abilities and skillset of the selected tribe member whenever the player chooses to view them.
The system must provide a means for the player to return to the game environment whenever the player wants to close the information.

C. Implementing and assigning tasks:
The system must display the implementation of the current tasks. After the task is implemented, the system must modify the game environment and notify the player of the next challenges.
The system must provide a means for the player to assign tasks to the tribe members. After the player assigns tasks, the system should display the modified tasks and skillset of each of the tribe members.

D. Artificial Intelligence:
The AI must perform the simulation of implemented and assigned tasks; it must also decide how these tasks solve the game challenges or affect the game environment. The AI must create the reports of the simulation for the use of the system. These reports should include the details of the challenges (or goals of the particular level) that are solved by the implemented tasks.
The system must compute the progress of the level based on the simulation reports. If all goals of the level are met, the system must display the end-of-level quiz; else the system must notify the player of the next challenges.
The AI must constantly monitor the safety levels of the tribe. If the safety level is high for a considerable amount of time (value above 95 for 2 minutes), the AI must add obstacles for the tribe to maintain the competitiveness of the game.

E. End-of-level Quiz:

The system must confirm whether the player has completed all the goals of the particular level before displaying the end-of-level quiz. The system must provide a means for the player to answer the quiz. The system should store the player responses for the quiz.

The system should compare the player responses with the actual answers and compute the score (on a scale of 0 to 100). If the score is above a satisfactory level (90), the system should progress the player to the next level; else the system must display a message informing the user of the low score and restart the current level.

F. Multiplayer mode:

When the player chooses the multiplayer mode, the system should search for available external players and switch to multiplayer mode if they are available.

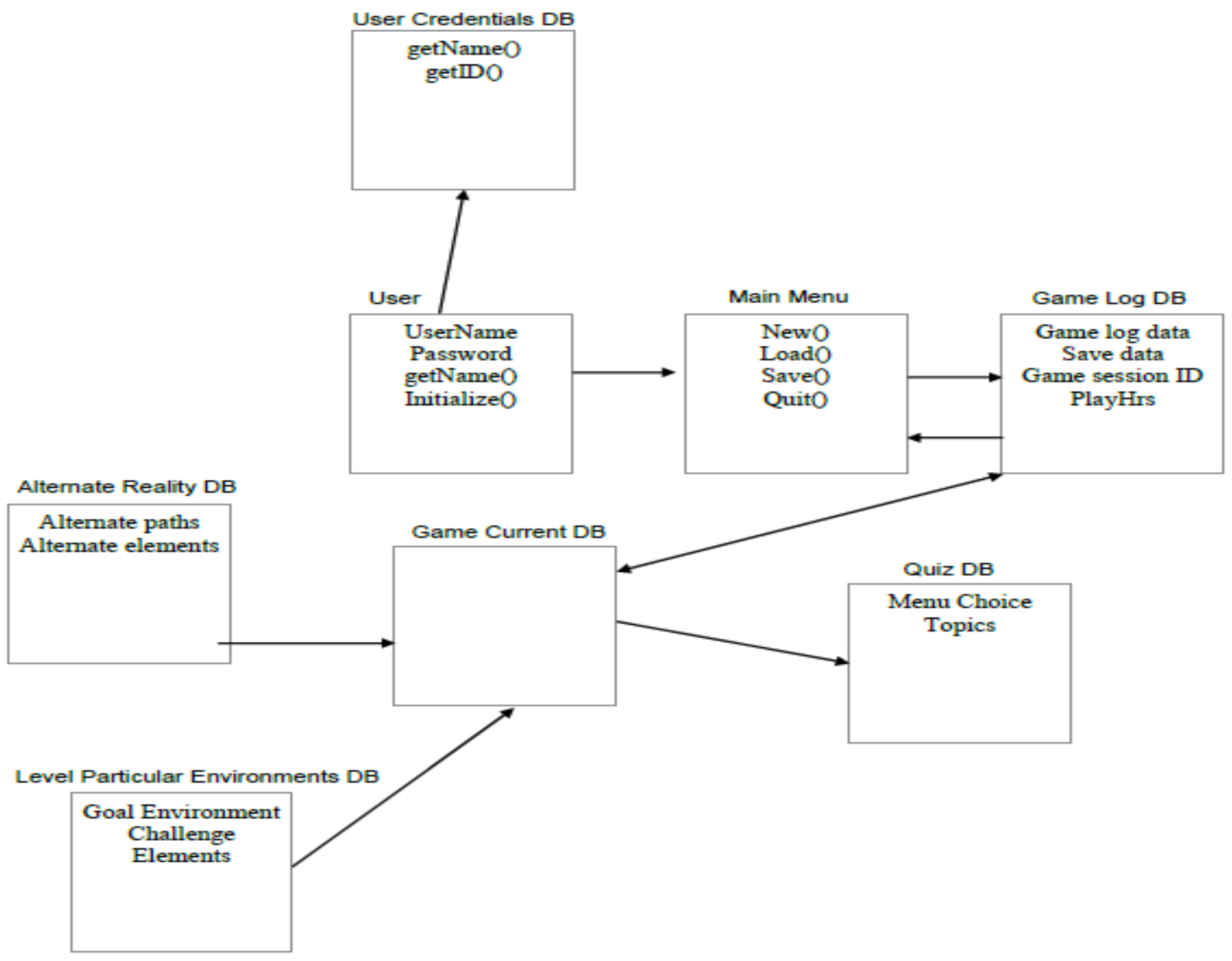
The system must compare the skillset of the player with the external players within the network and compile a list of compatible external players. The system must refresh this list every 30 minutes.

The system must display the list of compatible external players and the detailed information (such as the highest played level, skillset, number of tribe members, scores) of each of the external players.

The system should be able to send an invite to the external player to join the multiplayer game. The system should check the responses from the external players and display the list of external players who accepted the invitation from the player. This list should also include the external players who got their invitation accepted by the player. The system must provide a means for the player to choose between the two options – either collaborate or fight with a particular external player. If the responses of both parties are same, the system must display the multiplayer game environment; else the system must prompt the player to pick another external player or to change the response.

11 Data Requirements

The diagram shows the following databases – User Credential DB, Game Log DB, Alternate Reality DB, Level Particular Environment DB, Quiz DB, Game Current DB. The User Credential DB contains all the user accounts information. The Game Log DB contains save point information, hours played, progress of game play. Quiz DB consists of questions, topics and options for the quiz questions. The Alternate Reality DB contains the alternate paths and the alternate elements that should be simulated by the AI. The Level Particular Environment DB contains the goal environment and the challenge specific elements. The next is the Game Current DB, which contains the data fetched from the other databases that is currently used in the game. The user logs in with username and password and is directed to the main menu having functions such as New, Load Game, Save Game and Quit Game. Once an option is chosen, the required data is fetched from the other databases into the Game Current DB.



12 Performance Requirements

12a Speed and Latency Requirements

When it comes to online games, lag is a major issue. When the user is playing the game the number of frames per second (FPS) displayed to the user in high resolution should be 24FPS for 90 % of the time and it should not fall below 18 FPS under any circumstances. If the bandwidth of the user falls below 1 MBPS then the graphics should be compromised to lower resolution graphics but not on the number of FPS.

For every action the user performs on the system such as player selection for a particular task, selecting weapons etc., the response time should be less than 1 sec for 90% of the time and it should not exceed more than 3 sec.

Resource consumption at the end user's system also plays vital role. CPU utilization at the end user's PC should be less than 10% for 95% of the time. GPU (Graphical Processing Unit) utilization should be less than 75% for 95% of the time. Memory consumption at the end user level should be less than 15%.

12b Precision or Accuracy Requirements

The food and safety meters that indicate the level of food and safety respectively, must be represented in percentages and it must be accurate to one decimal.

After completing each level, the user will complete a quiz and the score calculation needs to be accurate and the average of the scores in quiz and scores earned at each level should be accurate to 2 decimals.

As we are receiving all the historical information from PreHistoric Inc., it must be accurate and there should be not be any spelling mistakes while representing names of ages and any anachronisms in the inventions and discoveries such as invention of wheel, discovery of fire etc.

12c Capacity Requirements

When more number of users join the game, the game server should be able to accommodate all the users and should not compromise on the performance. The game server should be able to accommodate at least one million users without any performance degradation for 99% of the time.

13 Dependability Requirements

13a Reliability Requirements

The program should not crash on any operating systems such as Windows, Mac-OSX and Linux for 99% of the time.

When the user is in multiplayer mode the game should not crash for 99.9% of the time.

The game server should fail no more than once in 10 million hours of game play.

If a crash occurs the user's game play data should not be lost and it must recoverable at any point of time.

13b Availability Requirements

The game should be available for the users to play 24 hours per day and 365 days per year. The game server should be available 99.9% of the time and in case of server maintenance, the server should resume functionality within 15 minutes and this should be during non-peak hours. Any longer down time must be notified at least 3 days in advance to all the users.

13c Robustness or Fault-Tolerance Requirements

In the unlikely event that the server crashes, the game play data of users should be backed up to a redundant server and it should take the place of actual server so that the system is available to the users.

13d Safety-Critical Requirements

Game resolution settings like gamma correction, brightness and the color contrast are set according to health standards and norms so that they do not affect the eyes of the users.

14 Maintainability and Supportability Requirements

14a Maintenance Requirements

Updates related to the maintenance of "Lithos" will be made available to the customers half yearly on the basis of any customer complaints, reported bugs specified to the company. A compilation of all these are considered and an official announcement from the company about the maintenance update is released before 3 months from the actual release of the maintenance update. All the planned bug fixes and customer complaint fixes which were mentioned in the announcement must be addressed in the update.

When there is a customer complaint regarding the improvement in the graphical user interface, it would take less than the mentioned time to release an update, while for a bug fix related update it takes more time when compared to graphical user interface problem. The release of a maintenance update depends on the severity of the problem.

The end users can send their feedback using a form provided by the contact us section in the game, all the forms are saved in a separate server maintained and accessed by the corresponding team responsible for maintenance updates.

14b Supportability Requirements

The game must be susceptible to modifications to accommodate the varying interests of the user which are collected through feedbacks. Apart from giving the users a help in textual format, there is a game tutorial to assist the user in the game play. So, the product in a way that is self-supporting.

In order to keep the cost of providing supportability to a minimum level, this game provides an online tracker which stores the incoming queries of the users. A game manager must be employed to address these concerns at least once a week. Once these queries have been addressed, they must be included in the help text if deemed crucial. The system should allow various other enhancements to be added without having to alter the existing source code to a greater extent.

14c Adaptability Requirements

The game can currently run on Windows (XP or higher), Mac OS X and Linux platforms. But the future releases would be compatible with phones and tablets in order to incorporate the ever changing user preferences.

Since "Lithos" is an online multiplayer game, it must run smoothly on any of the web browsers like that of Firefox or Google Chrome or Safari.

The adaptability of the game essentially depends on how well the game components and connectors inter-operate in any new environment. A specific metric which can be used to ensure this is given by Element Adaptability Index (EAI) which in turn contributes to the Architecture and Software Adaptability indices (AAI and SAI respectively). Upon comparing two different architectures for the same instance, the one which gives the maximum value for SAI is considered.

For the future addition of any software component, the same mechanism should be followed and the architecture which gives a higher value of SAI is agreed on.

14d Scalability or Extensibility Requirements

The game is expected to have at least 800 licensed schools and 700,000 users who are signed up by the time of the first release, there after it is thought to have a 20% monthly increase in the licensed users and 5% annual increase thereafter. The game servers can handle up to one million licenses for the game to run smoothly. Depending on the demand for the game later on, additional servers are deployed to maintain the users.

14e Longevity Requirements

The expected life time for the game is 6 years. The development, deployment, maintenance budget for the game is considered for this life time, further budget will be allocated for the game when the expected longevity varies.

15 Security Requirements

15a Access Requirements

This deals with the protection of customer's data. The game consists of both single player and multiplayer game. Both are accessed from online servers and the user's setting are attached to the game through user's account servers. The user creates an account with his personalization and it gets stored in the online server. It is difficult for any unauthorized access as the records are stored in a separate server not in connection with the game server.

The game account server should implement end to end encryption on both the login information and user's game play data. In case the user forgets his/her password, then a link will be sent via email to the user where he answers a list of security questions that he has set up during sign up process for successful change of password.

15b Integrity Requirements

The databases for providing the content to the game must be highly integrated into the game system, it has to be very secure in terms of introducing any unrelated data into it. All the data that is passed in and out of the database must be encrypted, so that no data can be stole, or changed. The game system must be designed such that the encrypted data can only be decrypted at the database or at the user end. Login function must be integrated in the game system whenever the data is requested or sent. There must be a backup system whenever there is a failure to the database system.

15c Privacy Requirements

The privacy requirements for single player and multiplayer game, the password required for accessing the users account is encrypted and the decryption key are kept private without any connection with the server. The user has to remember the password and input the decryption key whenever necessary. The server or any outsider can't access the password as it is encrypted. It is the responsibility of the user to keep the password safe as it might reveal the personal information.

15d Audit Requirements

The database system must store the user's record for a period of 3 years after the user's account has become inactive for the purpose of future legal concerns. After the period the details of the user's account will be deleted. The intention is that neither the company nor the user denies the fact that they have been associated with the product for a period of time or vice-versa in case of any legal issues.

15e Immunity Requirements

The game should have standard security measures where it considers spyware defenses against any popular attacks, and make sure the game can be defended against any unforeseen circumstances.

The connections between the end client and the game systems must be secure by making sure the connection is open for only sending valid data.

Both the server and the end client systems must be secured. The server must run on a virtual machine that works only for holding and sending valid data. The user end must be supported with an antivirus, as they are the most vulnerable people to be attacked.

16 Usability and Humanity Requirements

16a Ease of Use Requirements

The game should be both easy to learn and easy to use, especially for students from an age group of 12-16 because they form a major portion of our target audience. It should also be intuitive enough to ensure that once the user is familiarized with the flow of events, he should be able to accomplish the specified tasks even if he returns to play the game after a break of 3 months.

The interface must be such that the user does not have to build any prior knowledge or put extra efforts to use it.

All through the lifespan of the product, it must have a consistent interface; the terminology should not change, the design components and controls should stay in their usual locations and that all the prior functionalities retain their behavior upon the insertion of any new functions. Also it must be ensured that the user is provided with the help he seeks for, right at the time he seeks for it. This contributes to the usage of product efficiently.

An important requisite is that the system should allow customization of the GUI, according to the user preferences. These configuration changes made by the user should be saved in their user profile.

Feedback can be collected via surveys and questionnaires and they must be addressed before releasing the next update of the game. The overall satisfaction of the product can be estimated from these findings.

A good measure to investigate the total success of the product is to consider the number of times it has been downloaded , and also to monitor the number of users that are connected to the online game server at a particular instance of time.

16b Personalization and Internationalization Requirements

The game allows the user to customize his play wherein he can pick his own appearance, choose a color palette for the quiz and change fonts from the top right corner of the page. For every user, a profile is created which saves his personal settings. So, the next time he opens the game, it loads with these settings.

The option to choose a different language is available to the user at the time of installation, and the tutorial and rest of the instructions are presented to the user in that specific language. This enhances the user experience, particularly if this game is launched in a cultural setting different from the one it was built in initially.

16c Learning Requirements

A quick learning tutorial is integrated into the game which the user can choose to view after the game loads into the system by clicking on that option. Since this is a game where students form a major part of the active user set, one does not require any prior technical knowledge to understand the working of the product.

This initial tutorial is for a length of 10 minutes, which does not go over the whole game play in deep detail, but gives a fair overview of the entire game play in a language selected by the user. The text that’s provided to the user has a step by step description. The user can access this text during any point of play.

Over 80% of the users can learn to play the game by following the tutorial, while there might be over 10% of the users who go through the text to get a complete understanding of the game.

16d Understandability and Politeness Requirements

This game product does not use any complicated terms and phrases so that the user can easily comprehend how the game is advancing, the specific reason being that the target users are mainly children belonging to an age group of 12-16. It involves the use of certain meters and points which are self-explanatory. But it is assumed that the user has an understanding of the language in which the game is made to run.

Plus, it does not expect the user to have any kind of prior historical knowledge about human evolution, and the user can indeed proceed further in the game just by applying what he learnt in the previous levels. This helps in achieving the intended goal of the product and makes it more likeable.

All the technical intricacies, like how the system chooses a setting for the user, based on the number of points he has earned or the manner in which the counter strategies are selected by the AI engine must be hid from the user, assuring a simple and clear user experience.

16e Accessibility Requirements

The game has certain accessibility features which are directly built in the software. In order to specify a certain action/alert, this game does not use any color coding to make it accessible to people who cannot identify colors. Instead, an audio cue is provided for them, and a closed captioning is displayed in parallel to aid hearing impaired people.

To assist other forms of visual impair, the game does not use patterned backgrounds behind text or important graphics. The game also provides a keyboard access to all the menus and icons. The help text should contain the precise keyboard controls displayed lexicographically.

These features must be incorporated into the game to provide a uniform user experience and make the product accessible to a wider community of people.

16f User Documentation Requirements

The help section of the menu is going to contain the required user documentation. It covers all the information pertaining to how to play the game, the user controls and how to enable specific features in the game.

A user documentation is essential for a proper comprehension of the game and to guide the user through the rest of his play. An installation manual comes packaged with the software.

Along with the documentation for methodology, the user is guided with a tutorial at the beginning of the game and various hints to help him go further all throughout the play.

Whenever any updates are incorporated in the game, the documentation must be revised to enlist the updates. These modifications must be taken care by the developer. The more accurate and precise the documentation is, the more reliable the whole product becomes.

16g Training Requirements

The game itself assists the user/player by providing him with a basic tutorial of how to play the game. The game basically begins with a brief introductory story, which gives the user a pretty good idea about what the game is about. In addition to this, a textual form of "instructions" is available in the main menu of the game which makes the user aware of the intent of the game, how to proceed and the default controls set by the system for play.

No individual needs to be employed separately to train the users to use this product. Also, the user does not require an intense training since he would be guided the whole time, by hints and clues that pop up timely informing the user about his next ideal move.

17 Look and Feel Requirements

17a Appearance Requirements

As this is an interactive game intended for students, the game should be of vibrant interface. It should include the most immersive interface for the students to get engrossed in the game. Our aim is to motivate the players to enjoy and at the same time learn.

For Example, the game appearance should constitute high end graphics for the scenarios and well-structured guidelines for the game to be understood. Each and every age must be illustrated with different colors. In each and every age there must be a transition according to their classification.

The testers who test the game for any kind of faults, must also consider whether the game has an attractive and an effective appearance and indicate it in the test report.

The developers who try to render the game according to the test report must consider the appearance of the game and rework the appearance of the game accordingly.

17b Style Requirements

i. The characters in the game must be realistic 3d models of pre historic humans.

ii. The fonts used in the game shall be of the type ancient gothic style.

iii. The pointers shall be representing ancient weapons.

iv. The sounds included in the game shall not be of any modern music tones.

v. Menus in the game should be of manuscript style.

18 Operational and Environmental Requirements

18a Expected Physical Environment

The game shall be used by students in their computer labs. The game shall also be available on desktops, laptops and tablets, so the interface should be adjustable according to the screen size.

18b Requirements for Interfacing with Adjacent Systems

The game must operate on Windows operating system 7 and above, Mac-OSX 10.8 and above and Linux 12.4 and above.

18c Productization Requirements

The game shall be downloaded using a web link available on the Lithos page. Download and installation of a file not more than 2MB is needed, so any naïve user will be able to install the game by using the downloaded file and an active internet connection is required during the download process.

18d Release Requirements

New version of the game shall be released half yearly and every release will improve the user experience, graphics and game environments. New release shall not affect the previous game plays and user data.

Since the game has already been developed each release should cost no more than 10% of the total budget assigned for the initial development.

19 Cultural and Political Requirements

19a Cultural Requirements

The game shall be made available in languages English, Spanish, Chinese and Hindi.

The game shall not be offensive to any ethnic groups.

The game should simulate civilizations based on the region that the user selects. For example, the Indus valley civilization should be simulated if the user selects Asian part as the region.

19b Political Requirements

The requirements of the game should be finalized by the CEO of Lithos Inc and it should not be influenced by the board of directors who support the product financially. The CEO of Lithos Inc should focus his requirements based on the educational value of the game.

20 Legal Requirements

20a Compliance Requirements

The knowledge regarding the history and evolution of human being for inclusion in the game

are taken from Prehistoric Inc., As we are developing the game for a commercial release it is essential that Prehistoric Inc. would provide us the complete copyrighted data for developing the game. The inclusion of any other data source would not be allowed as it is not copyrighted with the game.

The following are the compliance requirements that must be set:

i. The product shall avoid all copyright infringements.

ii. The developer shall take lawyers' opinion that the product does not break any laws.

iii. The product shall be developed following the agile development process

iv. Personal information shall be implemented so as to comply with the Data Protection Act.

It should be better to avoid any development teams to work on the game who had previous experience working for the same domain in any competitor companies in order to avoid intellectual theft lawsuits.

20b Standards Requirements

i. The product shall give proper references and citations.

ii. The product shall comply with MilSpec standards.

iii. The product shall comply with insurance industry standards.

iv. The product shall be developed according to SSADM standard development steps.

v. The product shall be developed in compliance with the Gaming Standards Association.

vi. The product shall be according to the standards development organization advancement act of 2004.

vii. The product shall be in compliance with the Federal Trade Commission (FTC) agency of the U.S. federal government.

viii. The game will comply with the ESRB's guidelines for earning a E rating.

III Design

21 System Design

21a Design goals

The main goal of the game is to impart knowledge about prehistoric humans. The design of the game should strongly support the above statement by maintaining the quiz difficulty level and accuracy in the quiz questions. The design of the tutorials at the beginning of each level should be intuitive and highly informative with images, animations and rich text.

Look and feel of the game should resemble ancient times, to make the user experience interesting and engaging. But the terms used in the game menus and controls should resemble the present day games to enhance the user experience.

The components used in the game should be designed such that they are light weight and do not hinder the speed of the game. In case of poor bandwidth, the graphics resolution should be lowered but the speed of the game should be optimal.

Game score should be calculated asynchronously, so that there is no delay at the end of each level for score calculation.

22 Current Software Architecture

Present day games use the generic game engine software framework which consists of five different components – the main game program, the rendering engine, audio engine, physics engine and artificial intelligence. This basic software framework is provided by many vendors of which appropriate framework for the game is chosen by developers.

23 Proposed Software Architecture

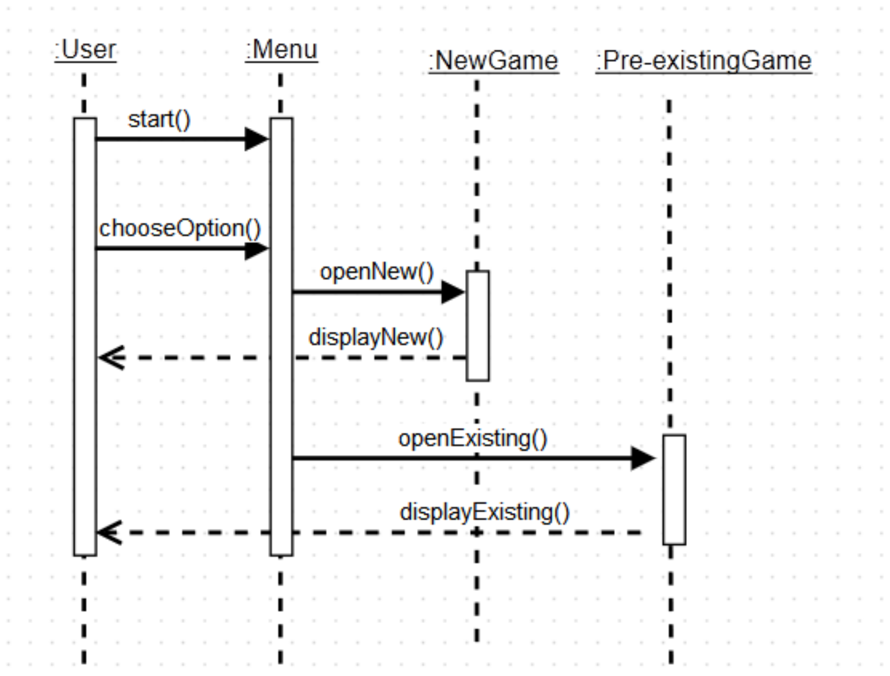
23a Overview

Lithos will use game engine software framework provided by UNITY. This UNITY game engine also contains the same five components (the main game program, the rendering engine, audio engine, physics engine and artificial intelligence) but UNITY allows for better code architecture by facilitating efficient code architectures like loose coupling, single responsibility principle, dependency injections, anti-patterns, singleton and manager classes.

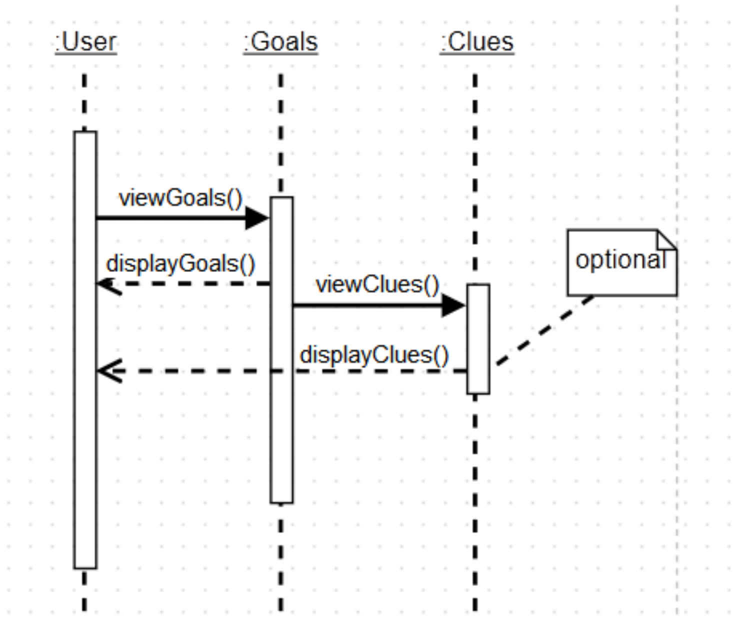
23b Class Diagrams

Refer to the figure in Data Requirement section (11 in Part II).

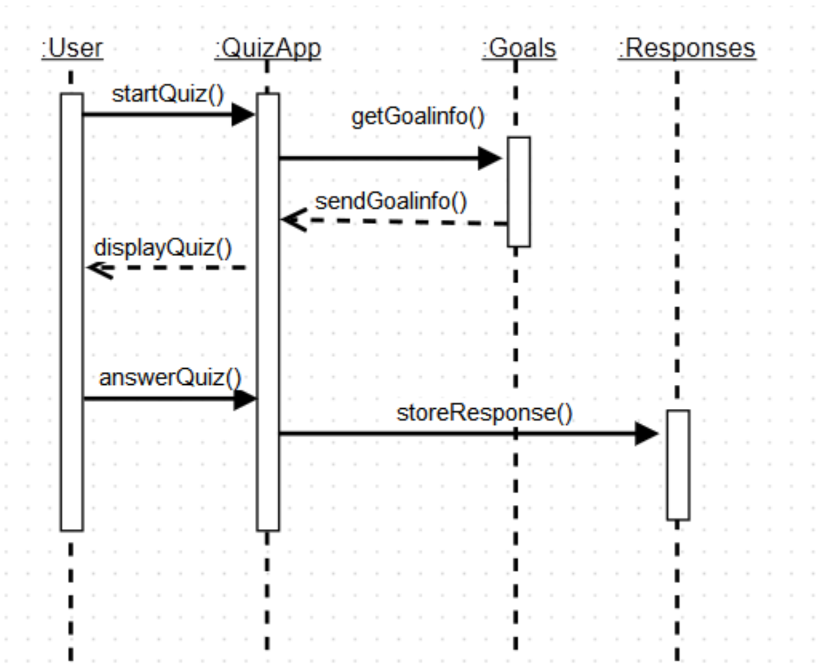
23c Dynamic Model



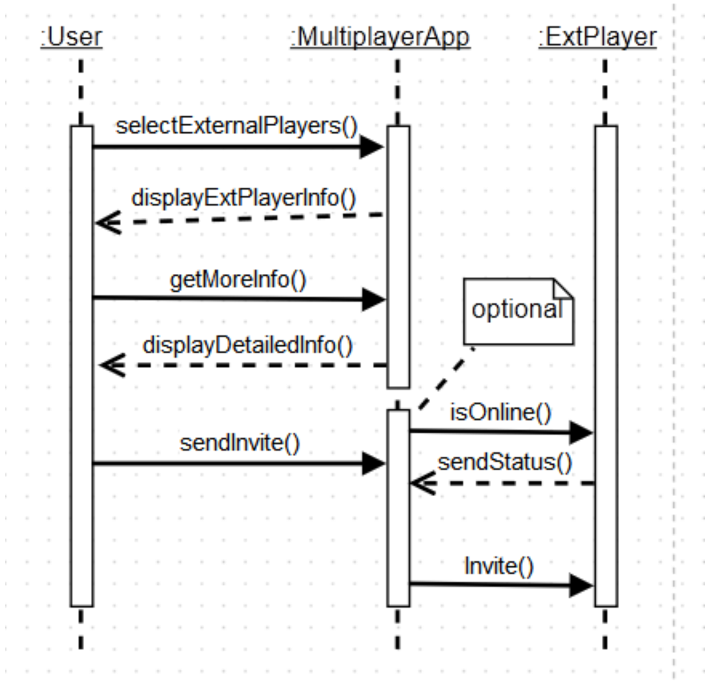
Sequence Diagram for Load Game



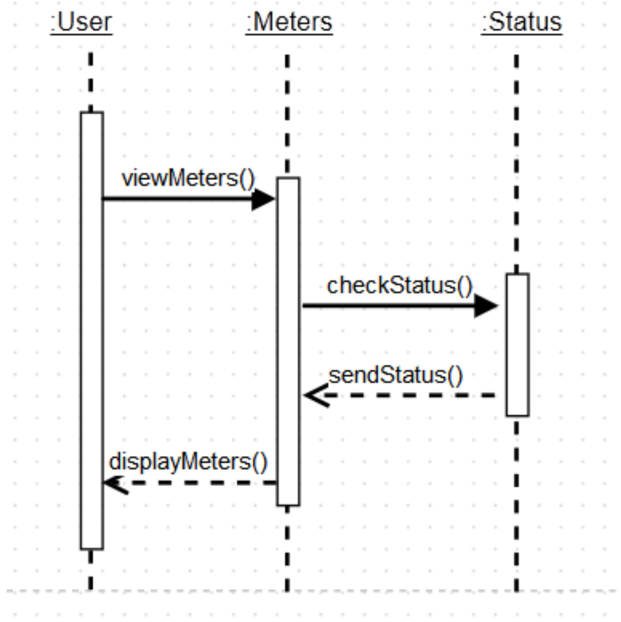
Sequence Diagram for View Goals and Clues



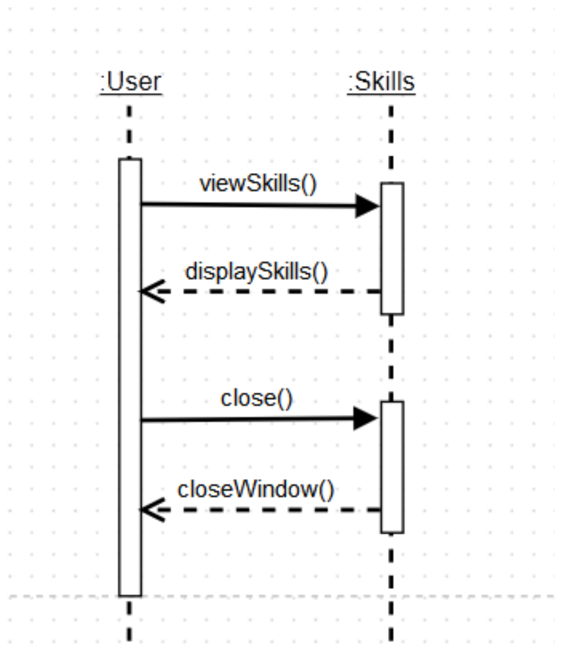
Sequence Diagram for Answer Quiz



Sequence Diagram for Select External Players



Sequence Diagram for Display Meters



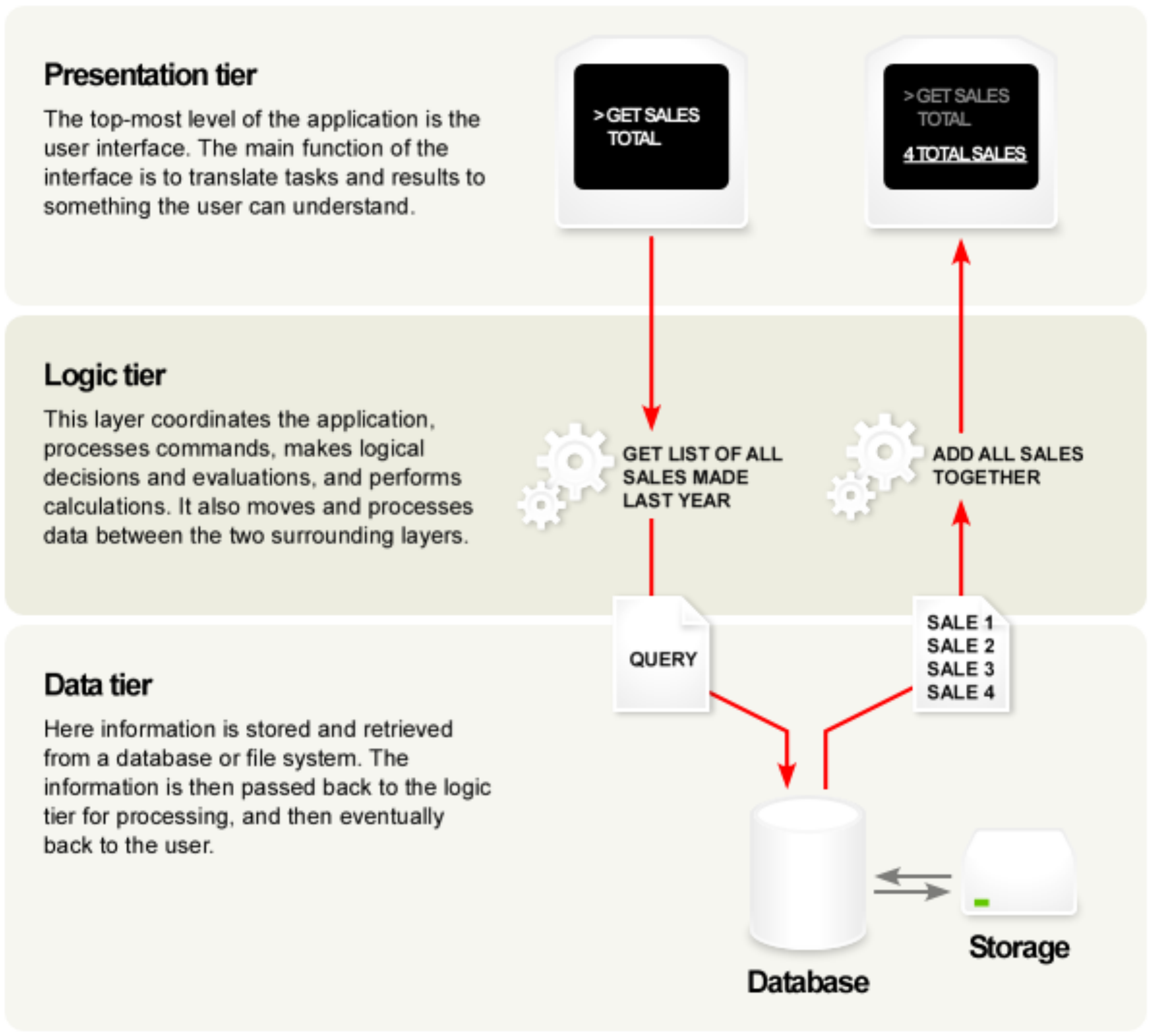
Sequence Diagram for View Skills

23d Subsystem Decomposition

Each subsystem is decomposed into the following layers: UI, Business logic, Networking layer, Database layer. Depending on the functionality of a subsystem, each layer is divided into multiple sub layers.

It is a good practice to have multiple layers for complex software application like Lithos. The N-tier architecture allows flexibility for the developers to add additional layers to the UI, business logic, database and networking. It enhances decoupling, reusability and makes independent module development easier. It will also restrict the functions in other layers from directly accessing the Database, thereby improving the protection from accessing the database from UI (SQL injection attack).

Example:

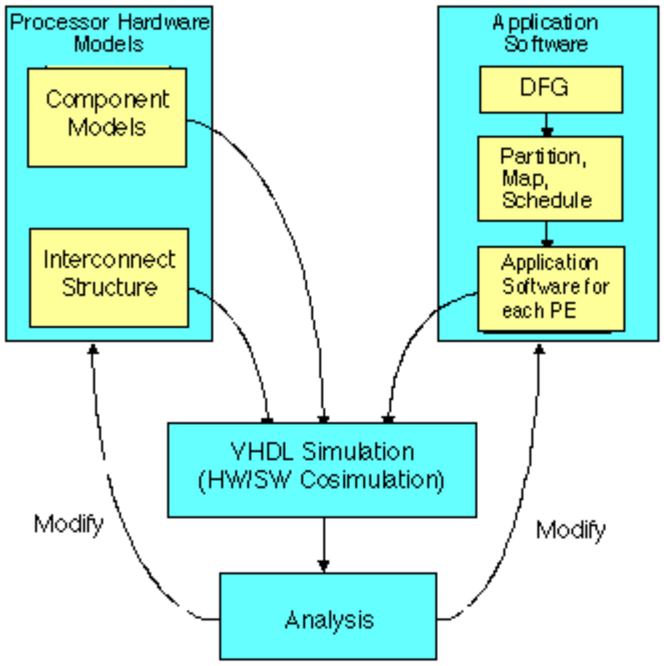

Presentation tier
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.
>GET SALES TOTAL
>GET SALES TOTAL
4 TOTAL SALES
Logic tier
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.
GET LIST OF ALL SALES MADE LAST YEAR
ADD ALL SALES TOGETHER
QUERY
SALE 1
SALE 2
SALE 3
SALE 4
Data tier
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
Database
Storage

23e Hardware/software Mapping

Lithos is a desktop application that will have to run on all major operating systems. Users PC should have at least 1 GB RAM as well as requirements that pertain to clock speedf 2.4 GHz (number of CPU Instructions per second) and 1 GB on-board graphics card.

To develop the software efficiently, knowing the target platform is essential - in this case, a PC. The code should be optimized and tailored for the target platform - therefore deciding on the platform early in the development process is important. The system should be designed in a way that it can be ported relatively easily to the mobile platform (Hardware/software wise) since it is very popular for casual users such as high school students.

The game should run as fast as possible - multi-threading should be used to achieve it. The minimum user PC requirement is a dual core 1 GB RAM so that the game can use multiple cores in the RAM making the game to have enough resources to run faster. In order to utilize the advanced threading functionalities, high end languages like Java or C# can be used while designing the core game logic.



23f Data Dictionary

Data dictionary is the database with which the application interacts. It will contain game data, user data, user play data, status of the game and level history information. All these details are saved in multiple tables in the database and joined when necessary.

Lithos uses MySql as its database as it can be easily hosted on Amazon AWS. The application connects to the Amazon AWS instance for pulling or inserting details to the database. An active internet connection is required on the PC where the game is running, to synchronize remote information between the application and the DB.

A timestamp has to be maintained in all the tables to keep a track of the changes to the table. A sample table design looks in the following way:

Table Name: [:tableName]

Columns: Date (YYMMDD), UID, [OptionalCol1],[OptionalCol2],..

23g Persistent Data management

Game play data, user data, game data is stored in the database. User game play data is used to predict his/her strength on those historical topics which are shown in the respective levels. So this data should be handled in such a way that it is not being compromised by any attacks and it should not be modified by any anonymous users. This data should be retrievable even if the database goes down.

The game data should be persistent and it should not be modified by any users outside the system. It is the main reason for the success of the game. So it should be handled in a proper and secured way.

23h Access control and security

Keeping the data secured from malicious attacks and avoiding bugs that users could exploit to cheat in the game is very important in determining the success of the application. The N-tier system helps prevent direct access from the UI to the database.

User data should not be compromised in any way. The database must be persistent and a fault tolerance mechanism should be implemented. Security mechanisms should also be implemented to prevent data from being leaked. The motivation stems from the understanding of how a secure system is critical to the commercial success of the application. This also helps in preventing horrific situations such as law suits due to stolen user data.

Recently there are reports of few data breaches at stores such as the Target and Home Depot where credit card information is leaked. This should be prevented while developing the application as it can have disastrous implications.

The security mechanisms should be implemented in such a way that the performance of the game is not being compromised.

23i Global software control

Software control is also known as a QA procedure which ensures the quality of the software before it is being deployed to the client. Depending on the size of the development team, sometimes there is a separate group of engineers who only deal with QA. In a smaller environment, those who wrote the module will have to test it and ensure the quality of the output.

There is great motivation for doing QA on the software as well as the data that it outputs. The check is not only for correctness, but also for efficiency. The test for the networking protocols being used in the game is made to ensure that no packets are dropped, so that it avoids the lag in the game because of bugs in the protocol.

Automated tools and technologies are used for verifying the quality that can help us validate the results faster and get back to development.

Logistics
Support
Delivery
Development
Research
Analysis
Quality Assurance
Production
Testing
Planning
Baseline System
Design
Prototype

23j Boundary conditions

These are conditions that can happen unexpectedly and very rarely also known as edge cases. They include cases like shutting down the software, booting the software etc. The user will have to experience a boundary condition sooner or later therefore we have to include these event handlers in our system, and make sure that we have the fault tolerance mechanisms in place to ensure upkeep of the data in case of failure. Implementation of the boundary conditions should be done in a way that doesn't require the overhaul of the entire system. The system also always needs to be in a stable deterministic state regardless of user actions.

24 Subsystem services

There are a few subsystems in the application such as the database and the network protocols that are used to deliver data between the clients and the server. All the subsystems must interact with each other flawlessly for the application to work properly. The database is responsible for holding all user generated data in various tables that can be later queried and joined. The networking system is responsible for communicating to the game server as well as the the cloud database. Defining the subsystems interaction is critical to creating an efficient application. There must be an efficient flow between subsystems without sacrificing the security of the application. The application should also be kept as extendable as possible so that additional components can be joined to the system when change is necessary.

25 User Interface

The UI should be simple, intuitive and responsive (fits to the screen the application runs on). The menus should be compact consisting more of buttons, selections and minimal text inputs. The UI should keep the player informed about the status of the game using information screens, popups and progress bars when background operations are running. The UI should not be cluttered, instead it should be easy to use.

26 Object Design

26a Object Design trade-offs

Important considerations in designing objects include:

- Graphics versus speed

  In order to give priority to speed, the game graphics can be compromised depending on the bandwidth.

- Centralized database vs Distributed query processing

  Instead of game instance querying multiple databases for information, a master database is designed which queries other databases. Now this makes the game instance query a single master DB rather than sending multiple requests to multiple databases.

26b Interface Documentation guidelines

The following general guidelines should be followed to maintain a uniform and clean interface:

- Methods should be verb phrases accompanied with a noun. Noun should start with a capital letter and verb should be in small letters. Eg., getUserId(), getGameLogData().
- Classes should be noun phrases and should begin with a capital letter.
- Instance variables should be in lower case.

- Constants should be in all capital letters.

There are two key interfaces in Lithos. Both of these must be clearly documented.

- Interface between game server and user's game instance. Client server architecture is used in this interface.
- Interface between game instances of external user and current user. Peer to peer architecture is used in this interface.

26c Packages

The game is divided into four packages:

- User interface package includes the classes related to menus, controls and their functionalities.
- Unity 3D package includes the classes related to rendering the graphics, physics and animations required for developing and running the game.
- User data package includes classes related to fetching user data like his Id, name, Game log data etc.
- Game content package includes classes for loading the game quizzes, level particular challenges, tutorial content etc.

26d Class Interfaces

i) Input Interface → classes in UI package

ii) Display Interface → classes in UNITY 3D engine

iii) Display Interface → classes in UI package

IV Test Plans

27 Features to be tested / not to be tested

For making Lithos a successful educational game, testing must be divided into three phases such as single player mode, multiplayer mode and the UI response of the game. The above three needs to be tested using black box testing methodology and the integration of different sub systems and unit testing individual features needs to be done using white box testing methodology. The features that need to be tested are all requirements specified in this document.

As this an educational game, it involves engaging the user at all points of time. They depend on the player mode (single or multi) and also on the UI response. So all these features needs to be tested thoroughly, to ensure the player, a continuous engaging activity.

During testing, it is crucial to take into consideration, the multiple instances of the game which are created by the game server. There should not be any errors in score calculation and data collection of individual players on a large scale.

Example: When calculating scores for a particular user, the score of the user for a particular level, the points scored in the quiz and the score for the user's previous game play must be taken into consideration and store them against the user id in the database.

28 Pass/Fail Criteria

While testing the components, the teams responsible for it, should make sure that the single player mode, multiplayer mode and UI of the game must pass black box testing, while the unit tests and the integration tests pass the white box testing. The unit test should not be carried out by the person who is responsible for developing it. A small team of developers are chosen to inspect whether the code has met all the stated requirements before even a test is carried out.

29 Approach

All the black box testing is done using the UNITY game engine and the unit tests are carried out in the chosen local platform like C#, C++. The full configuration tests must be carried out in all distributions of the operation systems: Windows (7 & above), Mac OS X (10.8 & above), Linux(12.4 & above).

30 Suspension and resumption

The game should be confined to the fail-safe mechanism. All the possible states for suspension and resumption of the game are simulated and the system is tested for the recovery of the game play data in each and every case.

31 Testing materials (hardware / software requirements)

The blackbox testing requires UNITY game engine that runs a workstation with windows as an operating system. The unit and integration tests are performed on windows operating system that runs Visual Studio. For Full configuration testing we use separate machines which run native operating systems Windows, Mac OS X, Linux.

32 Test cases

Test Case Name	Description	Expected Results
Lithos-DownloadSoftware	User vists the Lithos Inc website and downloads the latest version of the game. Repeat this on all platforms and browsers.	User successfully downloads the software to his local system.
Lithos-InstallSoftware	User begins installing the product by following the sequence of steps given in the read me file of the software. Repeat this on platforms such as Windows, Mac-OSX and Linux.	Software should be installed in the system and game icon should be created in the location that the user specifies during installation.
Lithos-LoginScreen	User double clicks on the game icon.	System directs the user to the login screen and prompts the user to enter his credentials
Lithos- LoginValidateSuccess	User enters his credentials (both username and password) and clicks on the submit button.	If the validation is successful then the game loads and system displays the main menu.
Lithos-LoginValidateUnsuccessful	User incorrectly enters his credentials and clicks on the submit button.	System displays the appropriate message such as "Incorrect Username", "Incorrect password".

Lithos-LoginUsernameEmptyCheck	User leaves username field blank and clicks on submit button.	System displays the message "Enter username".
Lithos-LoginPasswordEmptyCheck	User leaves password field blank and clicks on submit button.	System displays the message "Enter Password".
Lithos-LoginUsernamePasswordEmptyCheck	User leaves both username and password field blank and clicks on submit button.	System displays the message "Enter Credentials".
Lithos-UserRegistrationSuccessful	User enters registration data and clicks on submit.	System creates a new record for the user in the database.
Lithos-UserRegistrationUnSuccessful	User misses some of required fields in registration data and clicks on submit.	System should not create a new record for the user in the database.
Lithos-StartNewGame	User clicks on the new game option on main menu.	System displays all the completed levels and next level that the user can play.
Lithos-StartExistingGame	User clicks on the load saved game option on main menu.	System displays the pre-existing game where the user left in the last game play.
Lithos-ViewGoals	User clicks on the "view goals and clues" option in the game screen.	System displays the goals of the current level and provides the player with relevant clues according to the challenges of the level in a window.
Lithos-CloseGoalsWindow	User clicks on the close button in the goals and clues window.	System should close the goals and clues window and user should be able to resume his play.

Lithos-LevelCompletion	User achieves the level specific goals and clicks on continue game.	System should display corresponding quiz questions for that level.
Lithos-AnswerQuiz	User clicks on answer quiz button.	System will display a list of quiz question if he achieves the particular level goals.
Lithos-QuizCompletion	User answers all the quiz questions that were presented to him.	System displays the points earned by the user in the particular quiz.
Lithos-ScoreCalculation	User completes the level and answers the quiz corresponding to that level.	System displays the points earned by the user in that level and also total points obtained by the user till that level.
Lithos-QuizCompletionSuccess	User correctly answers 80% of the questions presented to him.	System displays the points earned and user can advance to the next level.
Lithos-QuizCompletionFailure	User fails to answer 80% of the questions presented to him correctly.	System displays the points earned and user cannot advance to the next level.
Lithos-AssignTasks	User assigns tasks to different tribe members in his team.	System must assign the tasks to respective tribe members according to user's intent.
Lithos-SimulateTasks	User assigns tasks to different tribe members in his team.	Skillset of the tribe members changes as the game advances.
Lithos-FoodMeterCheck	User collects food.	System must update the status quo of the food meter on a scale of 0 to 100.

Lithos-SafetyMeterCheck	User encounters any potential unsafe situation.	System must update the status quo of the safety meter on a scale of 0 to 100.
Lithos-SkillSetCheck	User clicks on view skill set option.	System must display tribe members list with their skills.
Lithos-MultiPlayerTest	User selects the multiplayer mode option.	System must display a message that the user has entered into the multiplayer mode.
Lithos-CompareSkillSet	User clicks on select external players in multiplayer mode.	System must populate a list of players with matching skill set.
Lithos-ViewExtPlayerDetails	User chooses to know more details about the external players.	System must display the details of the selected external player.
Lithos-SelectExternalPlayer	User selects a player from the list of compatible players.	System sends a request to the external player
Lithos-CollaborateOrFightOption	System displays a list of external players who have accepted the requests sent by the player and the requests accepted by the player himself.	Systems must show the options to fight or collaborate.
Lithos-CollaborateSelection	User must select the option to collaborate.	System must send the collaborate request to the selected external player.
Lithos-FightSelection	User must select the option to fight.	System must send the fight request to the selected external player.
Lithos-ExtPlayerCollaborateOrFightCheck	External player selects his/her option.	System must create the game environment for

		fight or collaborate if and only if both the players options are same.
Lithos-SafetyCheck	User is present in a safe situation for a relatively longer time.	System must impose threats to the user.
Lithos-SaveGame	User clicks on save game.	System must save the game play data to the gaming server.
Lithos-ExitGame	User clicks on exit game.	System saves the game and closes the application.

33 Testing schedule

Unit Testing	Duration: 2 days Begins: After each sprint is completed
Integration testing	Duration: 7 days Begins: 3 weeks before each release
Usability testing	Duration: 2 weeks Begins: 3 weeks before each release
Inspection Testing	During the entire development phase.

V Project Issues

34 Open Issues

There are a few open issues in the development of the game Lithos with no decisions on their solutions. They are described below:

i) For multiplayer game, decisions related to the security feature that has to be enabled for the login functionality are to be made. For the multiplayer mode, it is essential for the login functionality to have a security feature, which gives way for encryption and decryption of data. This requires large databases which leads to extra processing time.

ii) How the updates for the game should be released, as no patching mechanism is thought of for the updates in the game. The technical department must require certain amount of knowledge on how to patch code to already existing application. Thorough knowledge of C# is required for it. The design phase has not thought of the patching system, but it is essential for the game to have updates after the release.

35 Off-the-Shelf Solutions

35a Ready-Made Products

There are few ready-made solutions that would help creating the software. Some of them are:

- Unity 3D engine for creating game scenes and environments easily.
- MySQL database for storing the user's account data.

The purpose of mentioning Off-The-Shelf solutions is to make the development process of the game much faster. These are well developed softwares which decrease the development and research time.

There are many softwares which decrease the development and research time and speed up the development. The development team should look into different softwares which reduce the development time.

35b Reusable Components

The reusable components from UNITY are:

- Rendering Engine
- Physics Engine
- Audio Engine

35c Products That Can Be Copied

UNITY animation classes, animation packages and libraries can be copied and used in the development of Lithos.

36 New Problems

36a Effects on the Current Environment

The most important features of the game are data usage and graphics. The present software environment is all about memory usage. The game relies more on prevention of failure. The user who plays the game and uses the system for any memory hungry programs at the same time will have a bit of a problem as the game takes a lot of system's performance in the form of data and graphics. There should be proper implementation of memory drivers for Input/Output services of the game.

36b Effects on the Installed Systems

The interaction of the game with the other installed software is very limited. The game has its own predefined independent set of modules for its working. Tests must be performed on the game to test for any interaction of the game with installed software and operating system.

36c Potential User Problems

The potential user problem involves the usage of graphics and RAM. A user who plays the game with high graphical resolution will slow down the systems performance. Similarly, the game takes in a lot of memory allocation. Improper use by the user can limit the effectiveness of the game Lithos.

36d Limitations in the Anticipated Implementation Environment That May Inhibit the New Product

The present environment falls back on the graphical implementation, so it is difficult for the implementation of the graphics in the game in a normal environment. It puts in a lot of extra processing on the system for the implementation. Many steps are taken in the development phase to ensure that the game works accordingly to any specified system. Hence there might be some glitches while running the game in a normal environment when compared to a good system environment, but the game would run almost the same.

36e Follow-Up Problems

After the release of the game, there will be many competitors who will try to copy the game for different scenarios, it is very essential to maintain clients so as to render the game according to their requirements. Any unexpected changes in the educational system might bring a huge impact on the performance and business of the game. It is therefore required to take care of the game according to the changes.

37 Tasks

37a Project Planning

Lithos will be developed using Agile Development. The tasks with a higher priority will be chosen from the backlog and assigned to the team members in every sprint. Each release will consist of 4 sprints, each of which will span for a period of 2-3 months. Daily scrum meetings will be conducted to discuss the progress of the application and to resolve the issues encountered on the sprint tasks. At the end of every sprint, a working module of the product is built.

37b Planning of the Development Phases

Name of the Phase	**Operational Date**	**Operating Environment**
Version 0.1	8 months in	In- House
Version 0.5(beta)	20 months in	In - House/ On - Site
Version 1.0(release)	36 months in	On - Site

38 Migration to the New Product

38a Requirements for Migration to the New Product

There are no specific requirements for the game, as it teaches everything that is needed for it from the start of the game.

38b Data That Has to Be Modified or Translated for the New System

There is no necessity for any data to be modified or translated for the game as there are no similar kind of games to which adjustments are to be made.

39 Risks

The two major risks in the development process of the game are competition and data accuracy.

When the game is released, many competitors try to release similar games which are more advanced when compared to the game. It becomes very hard to grab our own share of business when more advanced games get released, eventually leading to losses. Hence it is quite essential to release updates according to the current scenario. It is also essential to involve all the features from the previous versions in the more advanced versions. There might be few legal problems the company would face while trying to incorporate those features.

The data providers for the game Lithos are the Prehistoric. Inc and few other sources such as books and universities. There can be few glitches in the data that has been provided, therefore it is very essential for the developers and data seekers in the company to crosscheck the data. The developers must also continually monitor the data distribution in order to quickly respond with the updates to be provided to the customers. If one source is considered unreliable, other sources must be in line to be considered.

40 Costs

The estimated budget for the initial release of the product is $850,000. Half of this amount is due upon successful completion of the User Acceptance Testing and the remaining is due upon the acceptance of the final product. The staffing requirements to maintain the multi-player server would come up to be around $40,000. Also, the expected cost to upgrade the server to accommodate the increasing number of users would be around $600/year.

41 Waiting Room

Some of the requirements which are collected in the requirement elicitation phase may not be included in the initial release due to time, cost or other such constraints. But it is highly essential that these should be integrated in the consequent releases of the game.

Releasing the Android/iOS versions of the game is a high priority requirement for the later developments and must be available in the 2nd release of the game.

The game should support user selection of game locations. In the initial release, the game play takes place in a location which is primarily a grassland. In the further releases, the terrains of hills, mountains and forestlands come into the picture. This too should be made available in the 2nd release of the game.

The game should give the the user a choice to select his own character. It should further incorporate an option where the user can monitor how the other tribes who are already defeated are currently stacking up and give him/her the power to take over them if chosen. This could be added in the next version of the game.

42 Ideas for Solutions

In order to provide more immersive experience to the user, the game is to be developed for 3D displays where the user can play the game with 3D glasses on. The solution for this problem is that there is off the shelf software that adds a third dimension to the existing 2D game.

The server should be up for 99.9% of the time. That means the server is up even when there are no requests which is a waste of energy. So we make use of a technique called soliloquy that allows the servers to go into a state like sleep and wake and serve when there is a request.

43 Project Retrospective

The requirements for the development of this game are formulated by considering a sample persona 'John' who is a high school student. Since the target audience mostly consists of students, the requirements are devised by taking into account the perspective of a potential user. This guaranteed that the expectations of the target audience were met and it gave a big push to the project.

The development of the project involved establishing short term goals which were accomplished successfully and an approach which was adopted is to cross-verify everything that was developed so far before moving on to the next stage. This ensured that we didn't sway away from our ultimate goal and that the intent of the game was not lost.

Coming up with an idea of a game which has a fair amount of historic as well as an educational value was itself a big challenge. In addition to this, more team collaboration would have helped during the integration of the individually developed modules into a successfully working product.

VI Glossary

DB: database

UNITY 3D: game engine software framework used for game development.

FPS: frame rate, Frames per second.

UX: User Experience

UI: User Interface

GPU: Graphics Processing Unit

Stone Age: This is the first stage in the prehistoric chronology of events.

Paleolithic Age: This is the first stage in the Stone Age that is further divided into 3 stages mentioned below.

Lower Paleolithic: This is the first stage in Paleolithic Age where early humans focused on basic needs like food and shelter, i.e., survival.

Middle Paleolithic: This is the second stage in Paleolithic Age where early humans focused on needs like Clothing and this stage also marks the discovery of fire.

Upper Paleolithic: This is the third stage in Paleolithic Age where early humans focused on strategic usage of fire for cooking, dealing with threatening animals and providing warmth.

Mesolithic Age: This is the second stage in the Stone Age where early humans learnt skills like stone carving and land technology.

Neolithic Age: This is the third stage in the Stone Age where early humans learnt techniques of agriculture and taming animals.

Bronze Age: This the second stage in the prehistoric chronology of events where skills like trading using barter system, invention of Potter's wheel and navigation got developed.

Iron Age: This is the third stage in the prehistoric chronology of events where the people have started using Iron to make weapons.

VII References / Bibliography

en.wikipedia.org

http://www.thinksupport.net/software_development.html

http://www.slideshare.net/integration-testing/importance-of-software-quality-assurance

http://www.unityninjas.com/code-architecture/unity3d-game-code-architecture/

Based on materials from Bruegge & DuToit, "Object Oriented Software Engineering" 3rd edition.