Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: Neophyte

Analysis Report

Ali Soyaslan, Oğuz Liv, Umut Aközs, Gülce Karaçal

Supervisor: Halil Altay Güvenir

Jury Members: Uğur Güdükbay and Hamdi Dibeklioğlu

Progress Report

March 19, 2018

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Contents

1.0 Introduction

In a rapidly digitizing world, having technical skills is very crucial since, nowadays; almost everything requires some form of programming. As technology has been developing, we have become more dependent on it and use various technologies to accomplish specific tasks in our daily lives. Technology is being implemented in almost every section of our lives and business structures. This is the reason why, many countries such as England, Singapore, Estonia and US have started programming education in early ages, because the sooner a person learns how to create programs, the stronger their problem solving abilities get. This education also amplifies their computational and analytical thinking skills. For instance, UK made the most ambitious attempt to get kids coding, with changes to the national curriculum in 2013. ICT – Information and Communications Technology – is out and replaced by a new "computing" curriculum including coding lessons for children as young as five [1]. Such knowledge is important not only to individual students' future career prospects, but also for their countries' economic competitiveness and the technology industry's ability to find qualified workers [2].

However, it appears that Turkey is a little belated to educate children about programming compared to other countries. According to International Computer and Information Literacy Study (ICILS), who conducted among students between 6-15 years from all over the world in 2013, it has been acknowledged that only 1% of students from Turkey have advanced computer knowledge. On the other hand, 35% of students from Korea, 34% of students from Australia and 33% of students from Poland have advanced knowledge about computers and programming [3]. In order to offer an effective and simple solution for this problem, the project Neophyte will be proposed. With Neophyte, we aim to teach children how to code while making them entertained by playing different kinds of games they like. Neophyte creates a platform where children can interact with each other in an exciting way and improve their programming skills.

In this report, we aim to provide an overall analysis of the system we will develop. First of all, the existing systems that are similar to ours, their qualities, and the missing features of the available systems are described. Then the details of our system are listed. Functional, non-functional, and pseudo requirements are presented. Afterwards, our system models are included.

Scenarios of Neophyte are explained in detail. These scenarios are generalized by common use case descriptions and the use case diagram is given. The object model and dynamic models of the system are also provided and explained. Finally, the screen mock-ups are included.

2.0 Current Systems

According to our market research, there exist applications offered in the market to teach children programming. These applications can be listed as follows:

- **Blockly:** This web based application is Google's simplified programming platform. This application helps kids in early ages of understanding programing concept. The help they get on this topic is about defining the algorithms in a simpler manner with graphics objects (jigsaw puzzles) [4].

- **Scratch:** This is another web based application from Massachusetts Institute of Technology (MIT), that helps children build games with, again, using jigsaw puzzle parts with different tasks. (i.e. a puzzle's job is to create a for loop and another puzzle is for boolean operations) This application is also good for building games for fun but it lacks a mission. Without a mission, children are pointlessly wandering around the application, trying to find a purpose for their appliances [5].

- **CodinGame:** This is the last web based application that we have found on our area of interest. This application is for more advanced coders, maybe around last years of high school or university age. This application is also useful but not for children [6].

Although these applications have similar functionalities to our system, Neophyte will have different and improved features than existing applications. First of all, Neophyte allows users to play the game in multiplayer mode. There will be tournaments based on completing time of the given tasks. Moreover, Neophyte provides children a platform in which they can follow each other and send direct messages. Therefore, this project will offer an improved and engaging environment where children may not only learn to program, but also have opportunities to be creative using programming.

3.0 Proposed System

In this section, a further description of Neophyte will be provided. Its features will be described in an overview. Afterwards, functional, non-functional and pseudo requirements of our system will be listed. Then, our proposed system models with use case diagrams, object and class diagrams, dynamic models and very early designs of the user interface will be presented.

3.1 Overview

Project Neophyte is a learning tool for children in elementary school and middle school ages. This tool helps children understand the concept of programming by teaching them the way of computer scientists and basics of simple algorithms. As it was discussed in the introduction, it is important for children to learn algorithm creation in early ages, as it will affect their problem solving skills and computational thinking skills.

This project aims to raise awareness among children about computer science topics by dragging their attention to these topics with games and fancy graphics. It is important for these games to be easy to understand as it should be challenging enough. Our baseline for these games will be psychological researches and pedagogical reports on child informatics.

Our application will consist of a number of small games (like flash games) and medium length scenarios that will contain many problems along the path of gameplay. For instance, the player will go through a parkour mission, and when players come to a point where they have to go up from stairs, there will be a sorting problem. If the player sorts the given container with success, the game will let the player go up from the stairs. This was of the more advanced examples of our project. Another example, which is easy enough for children, is that they will be tutored along the gameplay. First series of missions will let children conduct experiments on the system itself. They will be guided to create algorithms and make small game appliances. Then, according to their area of interest in real life, they will be directed to the games in these areas.

The language of this project is in Turkish and English for now. Normally, making these applications in another language than English is not good for many students, because the keywords of programming languages are all in English. Studying on foreign language basics are sometimes complicating for programmers. On the other hand, we are making this project for children in early ages of education, as our main mission is to endear coding to children, we will have no such apprehension on language. Plus, many children in Turkey still do not get proper English education in Elementary and Middle school.

Finally, this project also helps psychologists and penologists in many ways, such as, understanding a new way of children and a different way of communication with them. This project may take the initiative for a new research area as children will be motivated to use this system for both entertainment and educational purposes.

3.2 Functional Requirements

3.2.1 System Functionality Requirements

- The system should contain a help menu which explains how the game is played, where to and how to write code.
- The system should display the name and the total score of a user to other users.
- The system should pause the game and should stop all movements when the pause button is pressed.
- The system should be able to show credits.

3.2.2 User Functionality Requirements

- The user should be able to sign in via filling signup form with credentials.
- The user should be able to log in with registered e-mail address and password.

- The user should be able to edit personal information (name, e-mail address, password, location, company and school) in his/her profile whenever he/she wants.
- The user should be allowed to play single player or multiplayer.
- The user should be able to access the help menu that consists of information about how to play the game.
- The user should be able to pause the game whenever he/she wants.
- The user should be able to save the game whenever the game is paused.
- The user should be able to load the game whenever he/she wants.
- The user can toggle music on and off.
- The user can change game language between Turkish and English.
- The user should be able to view other user’s profiles consisting of their personal information.
- The user should be able to follow other users.
- The user should be able to send direct messages to other users.

3.3 Non-Functional Requirements

3.3.1 Usability

- The user interface must exhibit conceptual integrity and simplicity.
- The user interface should be user-friendly in this way; users can spend their time, enjoying programming rather than struggling to figure out how to play the game while writing code segments.

3.3.2 Performance

- The system should react to user’s input as soon as possible.
- Load time should be minimal.
- Controls should contain the minimum delay possible.

3.3.3 Extensibility

- The application should be able to include and support new features with ease in order to maintain the excitement and interest of the user, so it should be developed in a way that makes it easy to update.

3.3.4 Security

- The system should ensure security of personal data of the users.

3.3.5 Reliability

- In game compiler will be kept up-to-date in SDK terms.
- Coding assignments are strongly related with gameplay so that system integrity will be ensured.

3.4 Pseudo Requirements

- The application will have a cloud database and web-based application.
- Android application will be developed.
- Android’s AR library will be used for the AR functions.

3.5 System Models

In this section we will go in depth about the proposed systems analysis with the system models. The object model of the system is provided. The use case diagram is given and then scenarios which are generalized by common use case descriptions of Neophyte are explained in detail. Dynamic models of the system are also provided and explained. Finally, the screen mock-ups are included.

3.5.1 Scenarios

<u>Scenario 1</u>

Use Case: CreateNewAccount

Actors: Ryan

Entry Conditions:

- Ryan opens the app and is on Login page.

Exit Conditions:

- Ryan is currently looking at the Homepage.

Main Flow of Events:

1. Ryan presses "Sign Up" button.
2. Neophyte displays Create Account page and prompts necessary information needed to create an account.
3. Ryan enters his username as 'ryan123'.
4. Ryan enters his email as 'ryan-1234@gmail.com'.
5. Ryan selects and enters a password.
6. Ryan rewrites his password for confirmation.
7. Ryan presses 'Create Account' button
8. Neophyte verifies whether all necessary information is entered correctly.
9. Neophyte creates an account with the given information for Ryan.
10. Neophyte registers Ryan as User.

Alternate Flow:

9A. Ryan's entries are incomplete or inaccurate.

9A1. Neophyte warns Ryan about the missing or inaccurate entries.

9A2. Neophyte redirects Ryan to step 3.

<u>Scenario 2</u>

Use Case: LoginAsUser

Actors: Tom

Entry Conditions:

- Tom opens the app and is on Login Page.

Exit Conditions:

- Neophyte displays homepage or
- User authentication fails.

Main Flow of Events:

1. Tom enters his username as “tom1231”.
2. Tom enters his password.
3. Tom prefers Neophyte to not ask for his credentials again and presses “Login” button.
4. Neophyte checks whether or not given username and password are correct.
5. Neophyte verifies that the given credentials are correct.
6. Neophyte displays the Homepage.

Alternate Flow:

A. Tom enters wrong username and/or password.

5A. Neophyte fails to verify the username and password.

5A1. Neophyte displays a warning message.

5A2. Neophyte returns to the login screen.

<u>Scenario 3</u>

Use Case: UpdateProfileInformation

Actors: Brad

Entry Condition:

- Brad has just signed up and decides to edit his account information.

Exit Condition:

- Brad is currently on Settings page with his new settings.

Main Flow of Events:

1. Brad opens his account’s settings.

2. Brad changes his e-mail information that is taken during sign up process.

3. Brad adds his name and surname.

4. Brad saves the changes.

5. Success message is shown by the system.

<u>Scenario 4</u>

Use Case: SearchUser

Actors: Jennifer

Entry Condition:

- Jennifer opens Neophyte and is logged in with her account.

Exit Condition:

- Jennifer selects a user whose name is matched with what Jennifer typed in.

Main Flow of Events:

1. Jennifer selects User list from Neophyte.

2. Jennifer types in “Kate” and searches it from the list.

3. Neophyte displays the users whose name is matched with Jennifer’s keyword.

<u>Scenario 5</u>

Use Case: FollowUser

Actors: Jennifer, Kate

Entry Condition:

- Jennifer is already logged in and is currently looking to search results which are matched with the keyword she typed in.

Exit Condition:

- Jennifer follows Kate.

Main Flow of Events:

1. Jennifer selects a user whose name is Kate from the search results.

2. Neophyte displays Kate’s profile page.

3. Jennifer follows Kate by clicking on “Follow” button.

<u>Scenario 6</u>

Use Case: SendDirectMessage

Actors: Jennifer, Kate

Entry Condition:

- Jennifer is already logged in and is currently looking to Kate's profile.

Exit Condition:

- Jennifer sends direct message to Kate.

Main Flow of Events:

1. Jennifer clicks on "Send Direct Message" button.

2. Neophyte displays message screen.

3. Jennifer types her message on the space.

4. Jennifer clicks on "Send" button.

5. Neophyte sends the message to Kate.

<u>Scenario 7</u>

Use Case: StartNewGame

Actors: Will

Entry Condition:

- Will is on the Home Page and is currently looking at the list of games.

Exit Condition:

- Will starts the game.

Main Flow of Events:

1. Will selects a topic from the list consisting of science, math, sport etc.
2. Neophyte lists the games with the selected topic.
3. Will selects a game by clicking on its visual.
4. Neophyte displays a short description of the selected game.
5. Ryan presses "Start" button.
6. Neophyte starts the game.

<u>Scenario 8</u>

Use Case: SaveGame

Actors: Millie

Entry Condition:

- Millie is currently playing a game.

Exit Condition:

- Millie saves the current state of the game.

Main Flow of Events:

1. Millie chooses a game.

2. Neophyte saves the game’s state.

3. Neophyte displays the current game screen and asks Millie whether she wants to continue playing or to go back to the Home Page.

4. Millie chooses to go to the Home Page.

5. Neophyte displays the Home Page.

Alternate Flow:

A. Millie chooses to continue the game.

5A. Neophyte displays the current game screen.

<u>**Scenario 9**</u>

Use Case: LoadGame

Actors: Bob

Entry Condition:

- Bob is on the Home Page and is currently looking at the list of games.

Exit Condition:

- Saved version of the game is loaded.

Main Flow of Events:

1. Bob selects a game by clicking on its visual.

2. Neophyte displays a short description of the selected game.

3. Bob presses “Start” button.

4. Neophyte checks whether there is a saved state of the game or not.

5. Neophyte asks Bob whether he wants to load the saved version or he wants to start a new game.

6. Bob chooses to load the game.

7. Neophyte starts the saved version of the game.

Alternate Flow:

A. Bob chooses to start a new game.

7A. Neophyte displays the new game screen.

3.5.2 Use Case Model

3.5.2.1 Use Case Diagram

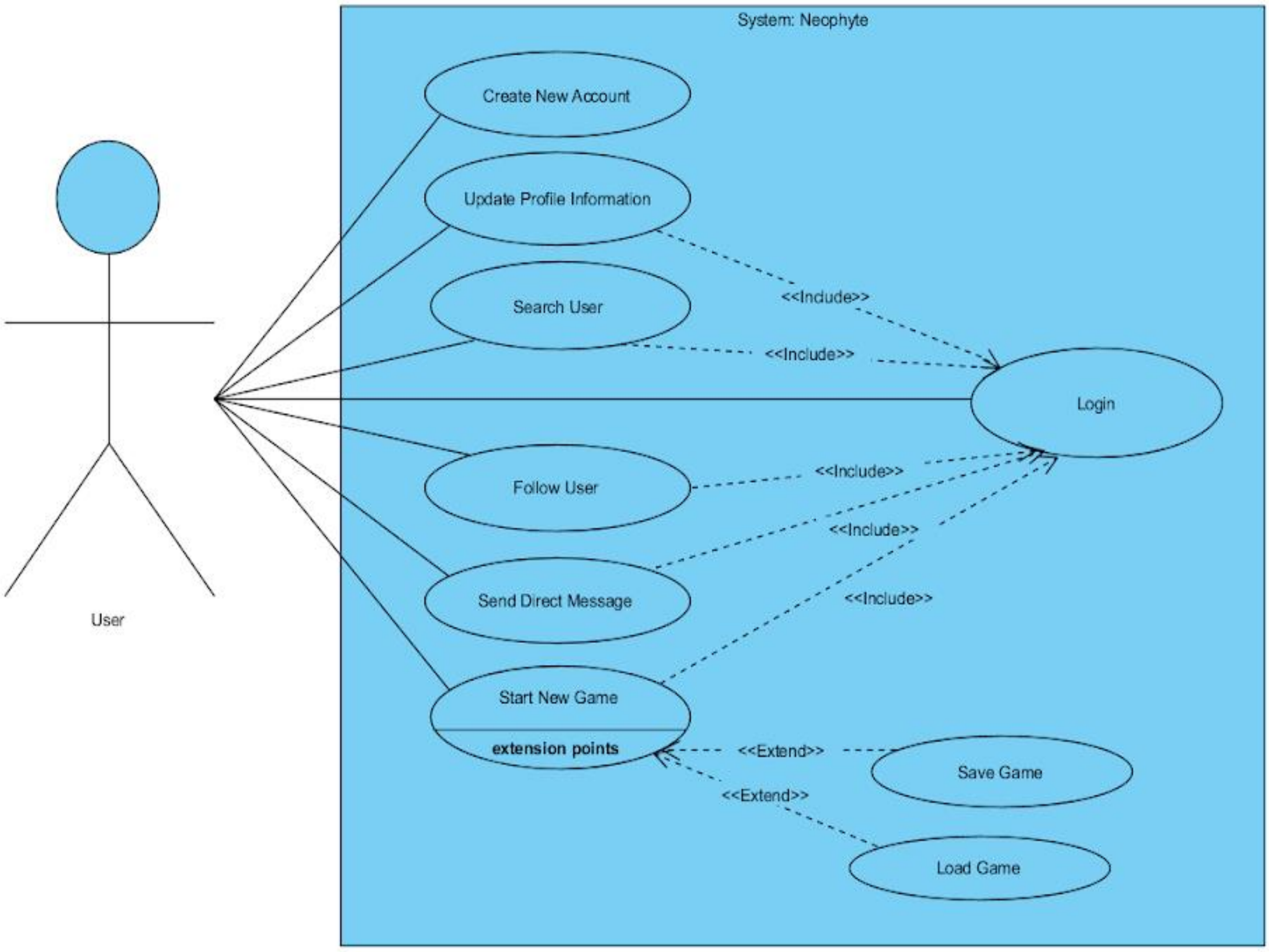


Figure 1: Use-Case Diagram

3.5.2.2 Use Case Descriptions

➢ **Create New Account:** User can create a new account by providing necessary information.

➢ **Login:** User can login to his/her account by typing his/her username or email and password.

➢ **Update Profile Information:** User can edit his account by changing the existing information or adding new information.

➢ **Search User:** User can search for another user by entering a keyword.

➢ **Follow User:** User can follow other users.

➢ **Send Direct Message:** Users can send direct messages to each other.

➢ **Start New Game:** User can start a new game after a preliminary description of the game.

➢ **Save Game:** User can save the game whenever she/he wants.

➢ **Load Game:** User can load a game which is saved before.

3.5.3 Object and Class Model

3.5.3.1 Class Diagram

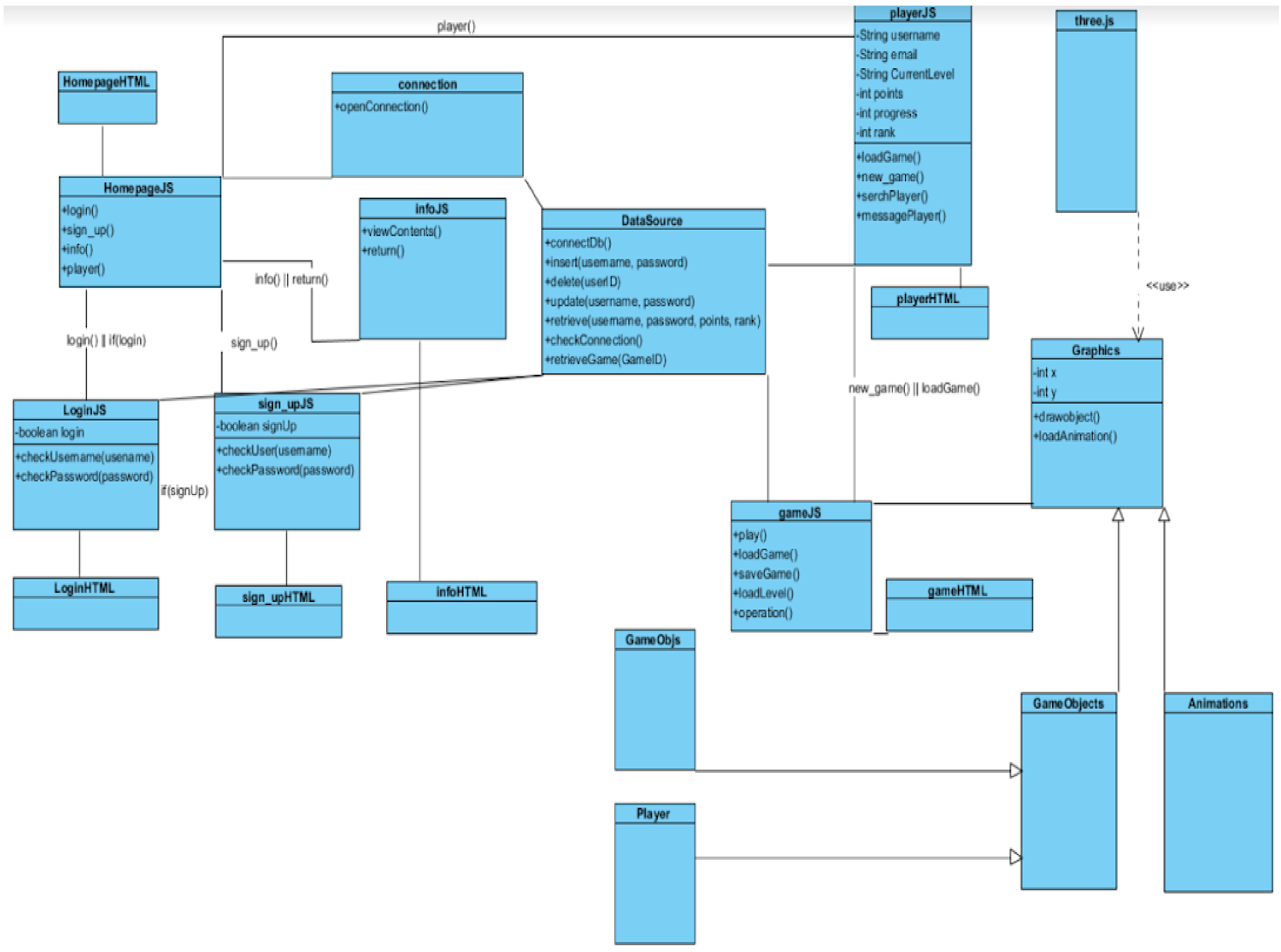


Figure 2: Class Diagram

3.5.3.2 Explanation of the Class Diagram

- **HomepageJS:** This class loads the first screen, login page. Also directs to other pages such as player page, login page, sign up page and info page. Moreover, interface of this class is created with HTML5 in HomepageHTML.

- **LoginJS:** This class checks username and password. It takes those values and checks database via Datasource class. If login data is true, then it directs user to homepage again. Interface of this class is created with HTML5 in LoginHTML.

- **Sign_UpJS:** This class helps non-user customers to register to database. It gets username and password value and checks database if password and username is available for database via Datasource object. If so, it directs new user to login page. Interface of this class is in Sign_UpHTML.

- **infoJS:** This class shows contents of credits, pays tribute to who worked on this project. Allows returning to homepage. Interface of this class in infoHTML.

- **DataSource:** This class manages database operations. When manipulation of database is inevitable, then other classes must use DataSource object to do required manipulations such as adding new user, updating user values.

- **Connection:** This class handles connection to server by Node.js.

- **PlayerJS:** This class is for player operations. Allows players to open new game or continue their saved game. Also provides messaging and searching for other users. Interface of this class is in PlayerHTML.

- **GameJS:** This class contains game logic. All game details and graphics are combined in this class. Interface of this this class is in GameHTML.

- **GraphicsJS:** This class contains all graphical details. Especially, it uses three.js library to create 3D graphics. It differentiates tools for game objects and for game animations.

- **AnimationJS:** It provides tools and procedures to create animations for specific purpose through game.

- **GameObjectJS:** It provides tools for game objects such as player's himsel/herself, other static graphics throughout the game.

3.5.4 Dynamic Models

In this section the dynamic models of the proposed system will be explained in detail.

3.5.4.1 Sequence Diagrams

Login Sequence Diagram

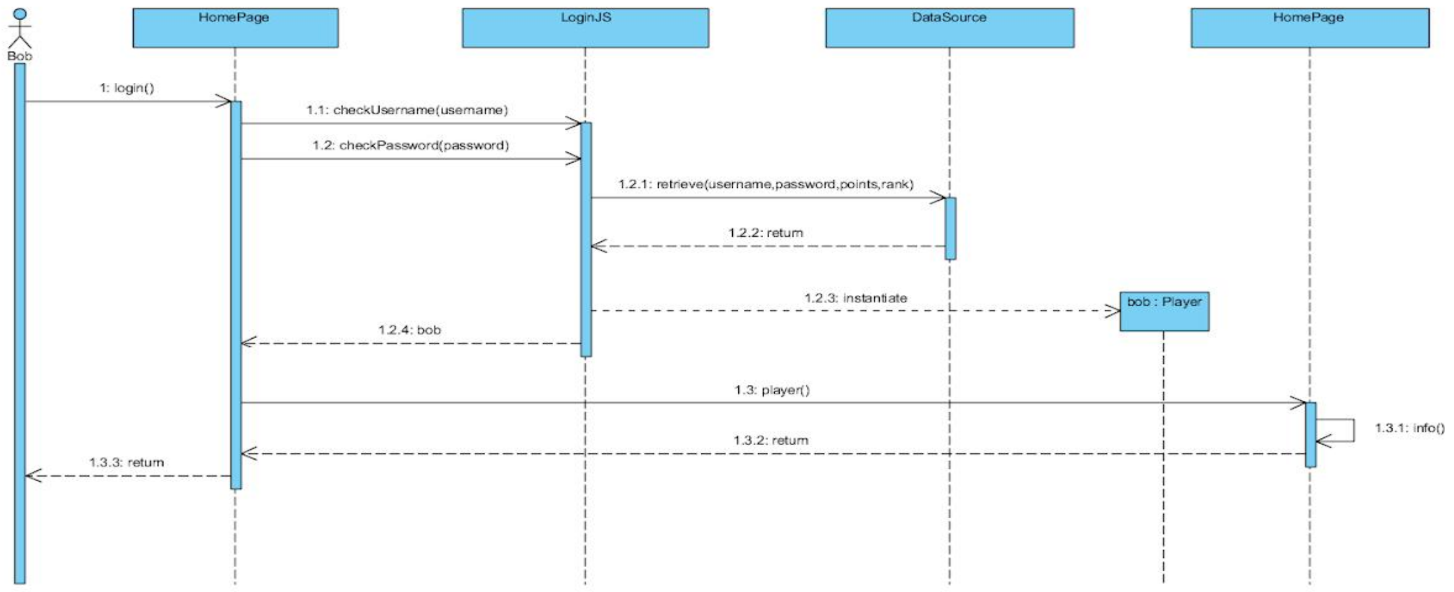


Figure 3: Login Sequence Diagram

The login sequence diagram shows the flow of execution of login process to Neophyte. First of all, Bob is prompted to enter his credentials, namely his username and password. Neophyte then verifies these credentials through LoginJS, a class which has the permission to manipulate user database. After validating credentials, LoginJS retrieve Bob's information from DataSource. Then, the Home Page will be displayed to the user.

Sign Up Sequence Diagram

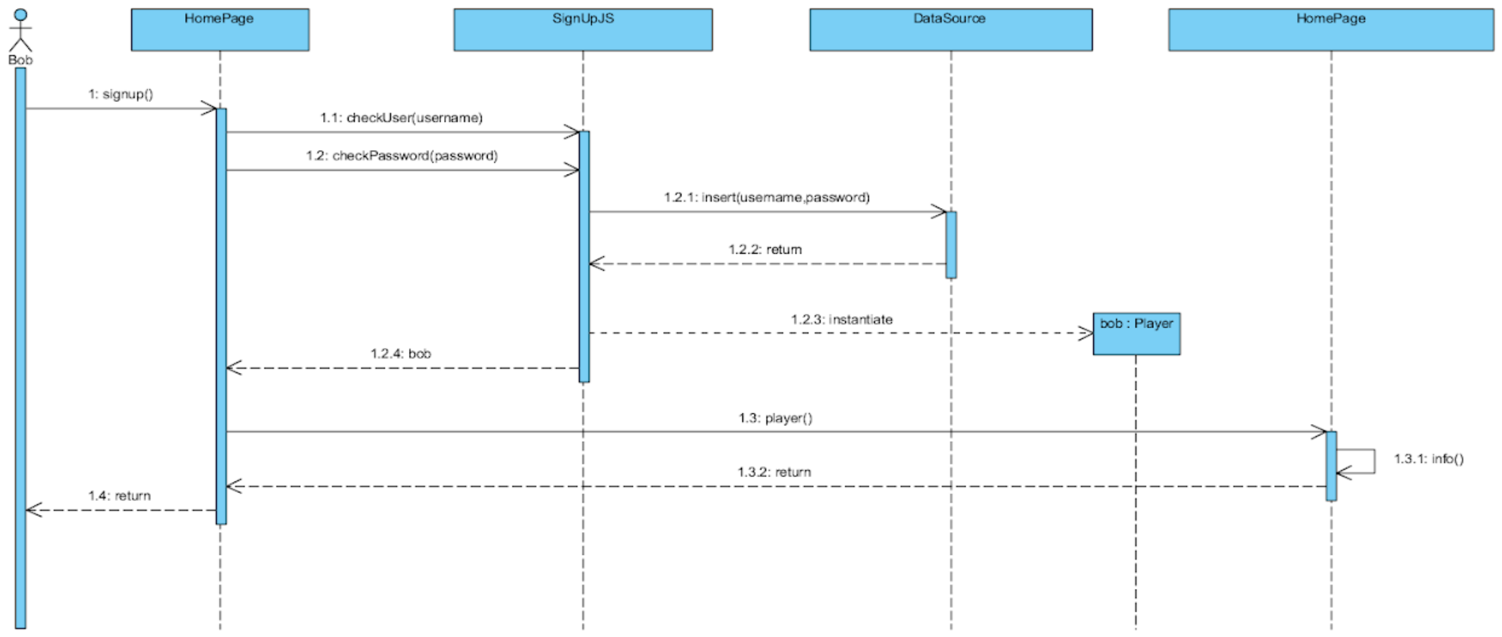


Figure 4: Sign Up Sequence Diagram

The above diagram shows the flow of execution of sign up process to Neophyte. First of all, Bob is prompted to enter his credentials, namely his username and password. Neophyte then verifies these credentials through SignUpJS, a class which has the permission to manipulate user database. After validating credentials, LoginJS insert new user to DataSource. Then, the Home Page will be displayed to the user.

Start New Game Sequence Diagram

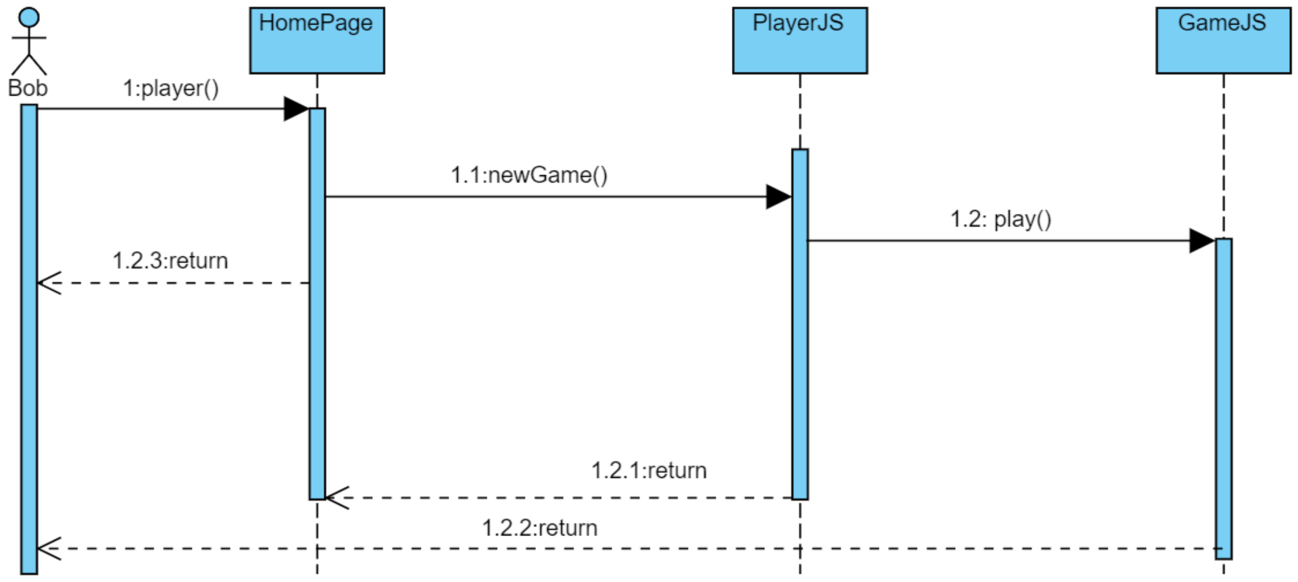


Figure 5: Start New Game Sequence Diagram

The above diagram shows the flow of execution of start new game process to Neophyte. After Bob is prompted to enter his credentials, namely his username and password, he can see New Game button in Home Page and if he clicks this button, PlayerJS class directs him to GameJS and he have to start new game by play() in GameJS. All this selection made by him returns to him as a messaging so that in Home Page, PlayerJS and GameJS give return message to him about what he did.

<u>Save Game Sequence Diagram</u>

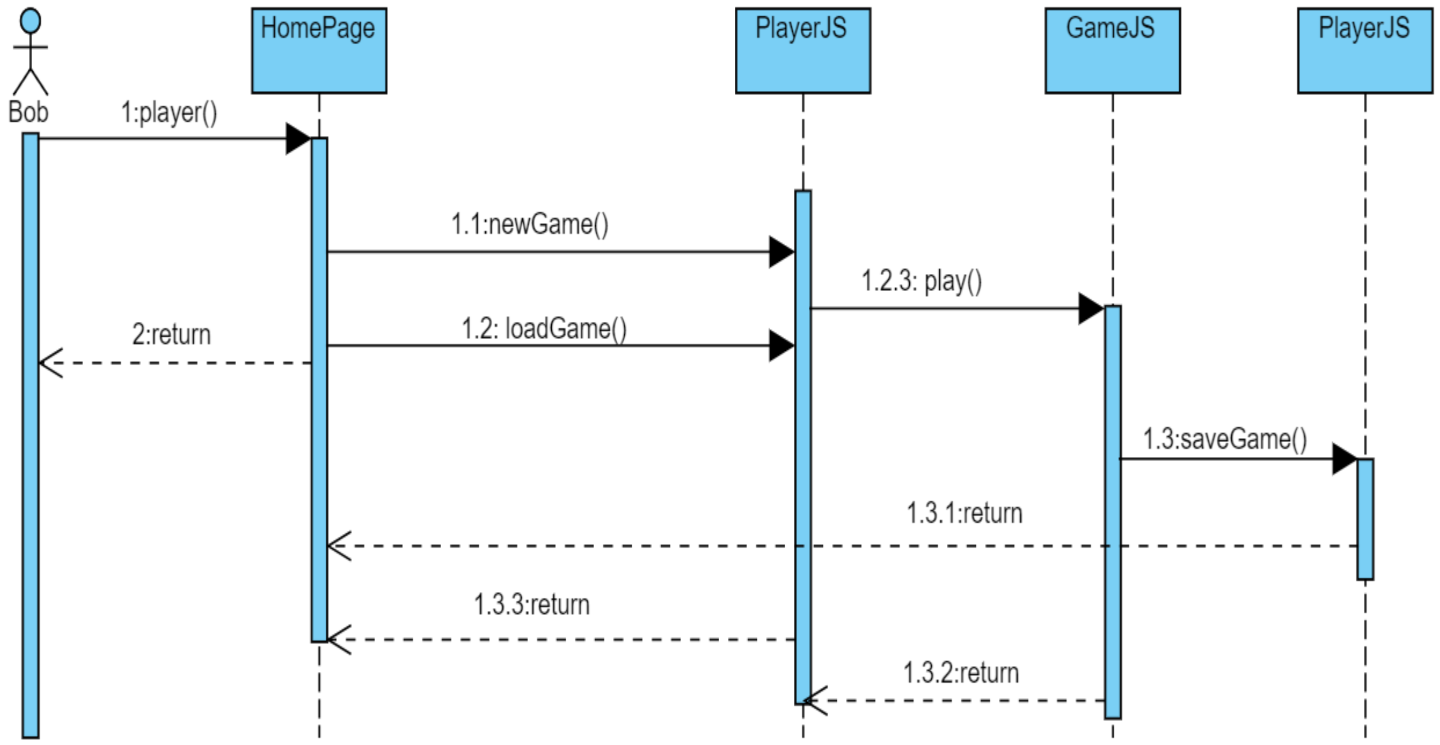


Figure 6: Save Game Sequence Diagram

The above diagram shows the flow of execution of save game process to Neophyte. After Bob started to a new game and he wants to save this game, he can click save button when he plays in GameJS. Previous of this process is the same with starting a new game or load game.

Load Game Sequence Diagram

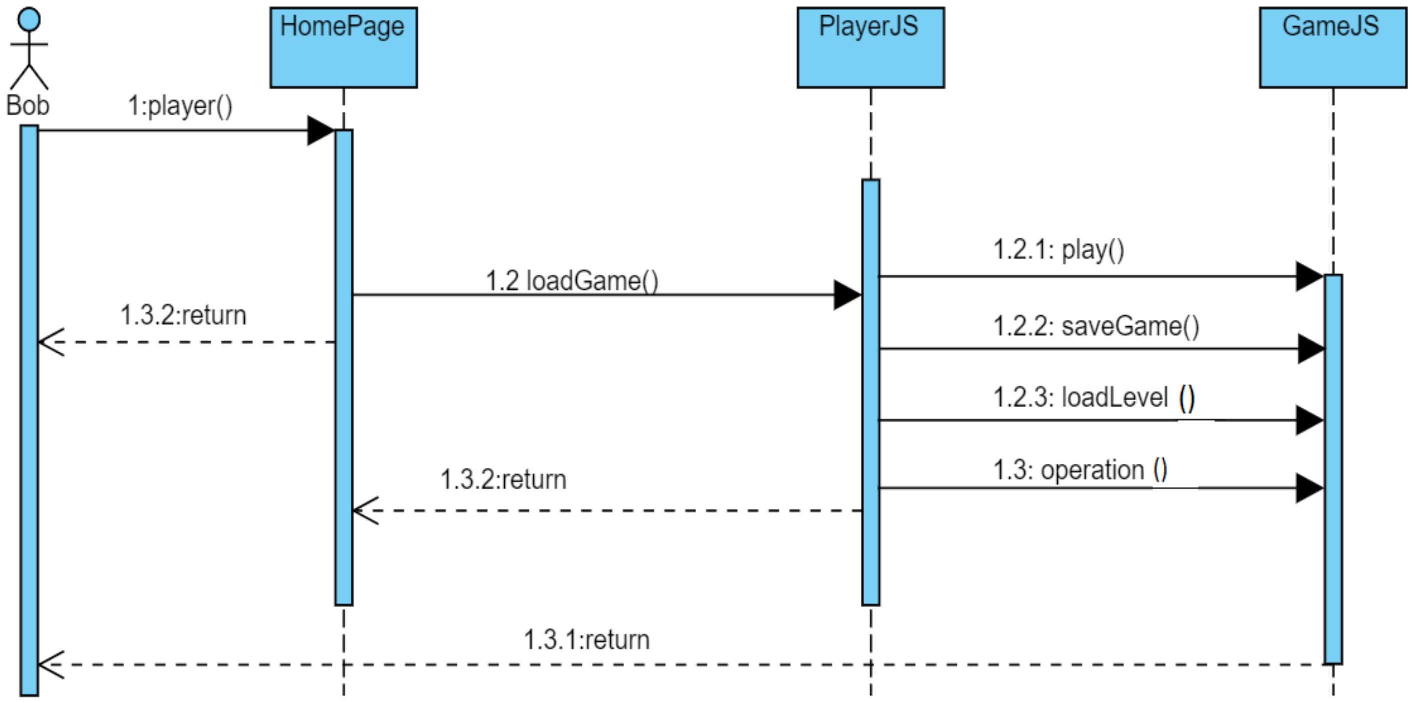


Figure 7: Load Game Sequence Diagram

The above diagram shows the flow of execution of load game process to Neophyte. After Bob saving a game and he wants to continue to play this game, he can click load button when he is in HomePage and this button direct him to PlayerJS and he can play, save and load level and some operations in this PlayerJS class.

3.5.5 User Interface

Login:

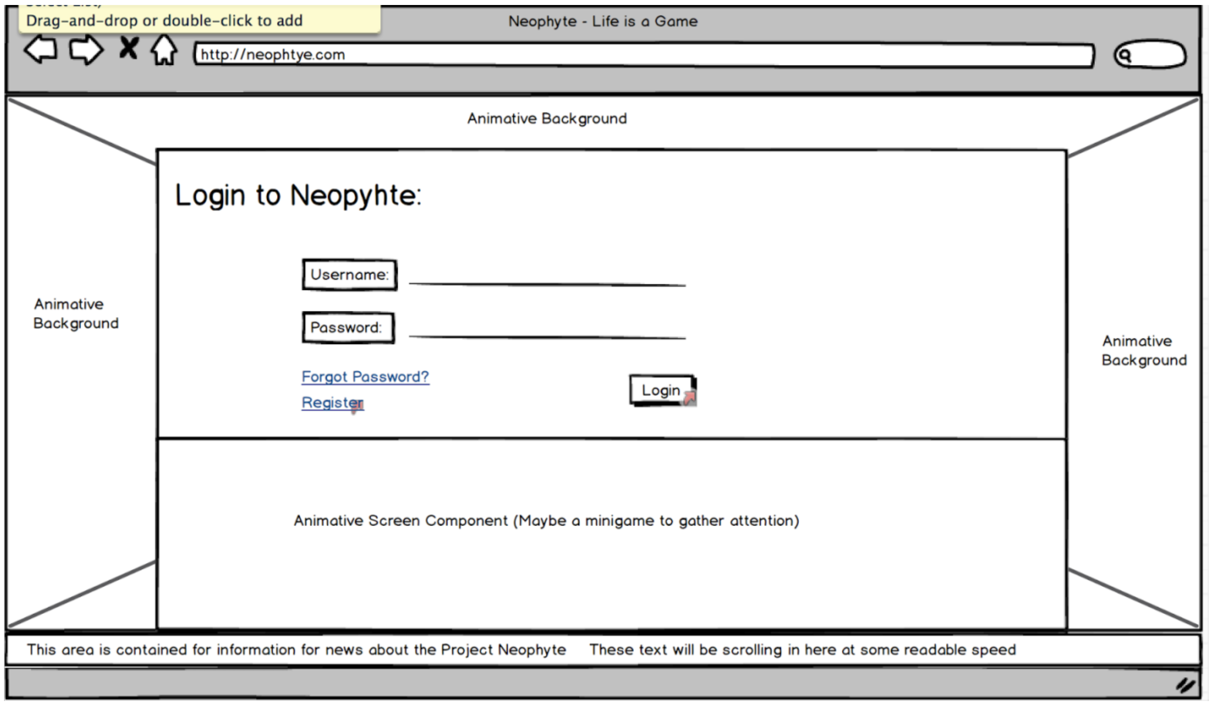

Drag-and-drop or double-click to add
Neophyte - Life is a Game
http://neophtye.com
Animative Background
Login to Neopyhte:
Username:
Password:
Forgot Password?
Register
Login
Animative Background
Animative Background
Animative Screen Component (Maybe a minigame to gather attention)
This area is contained for information for news about the Project Neophyte
These text will be scrolling in here at some readable speed

Figure 8: Login Screen

This page is the first page users will encounter. This page contains empty text boxes for users to type in their credentials for the system login. Their credentials will be controlled by the system and if their hashes match, then the system will login. The back-end system will work on a database for now but it may also work on Cloud according to the requirements.

<u>Sign Up:</u>

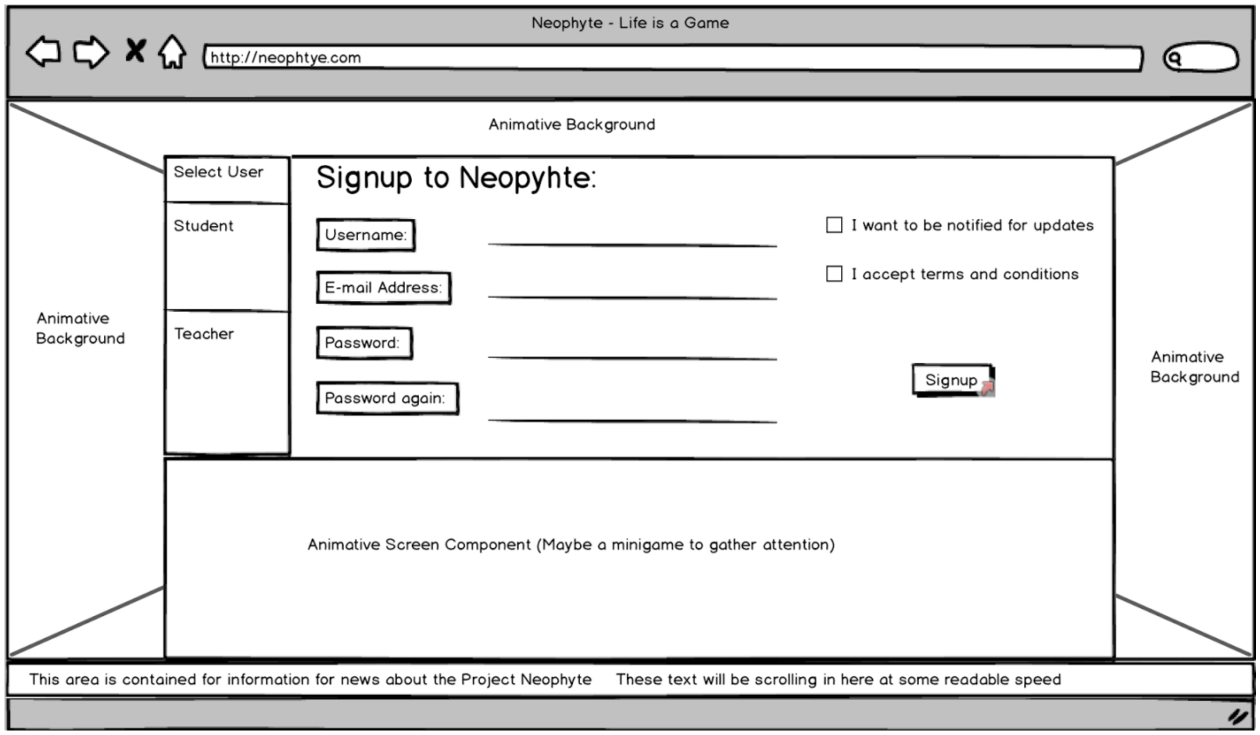

Neophyte - Life is a Game
http://neophtye.com
Animative Background
Select User
Student
Teacher
Signup to Neopyhte:
Username:
E-mail Address:
Password:
Password again:
I want to be notified for updates
I accept terms and conditions
Signup
Animative Background
Animative Background
Animative Screen Component (Maybe a minigame to gather attention)
This area is contained for information for news about the Project Neophyte
These text will be scrolling in here at some readable speed

Figure 9: Sign Up Screen

Signup page will demand an active e-mail address, a unique username and a password. These credentials are for people who bought it from retailers. There may also be another business field for this project, B2B. If this application is sold to schools or special study centers, then the system will require school or center name with some identification key.

Profile Page:

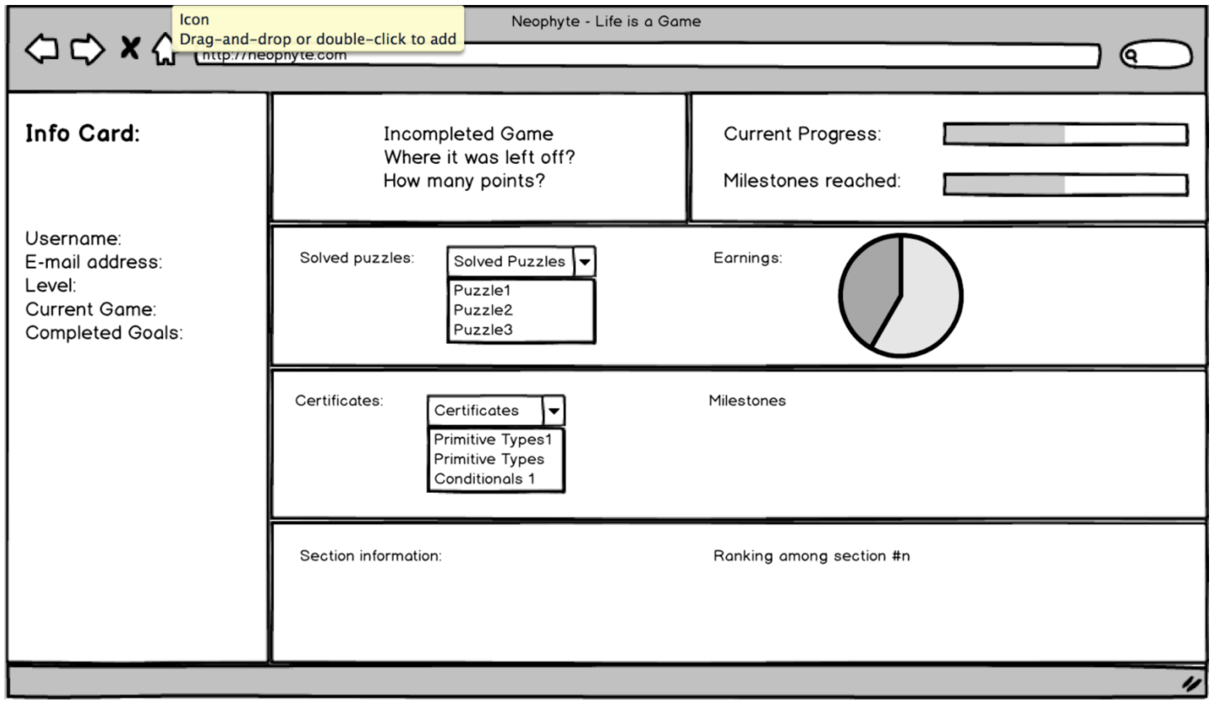
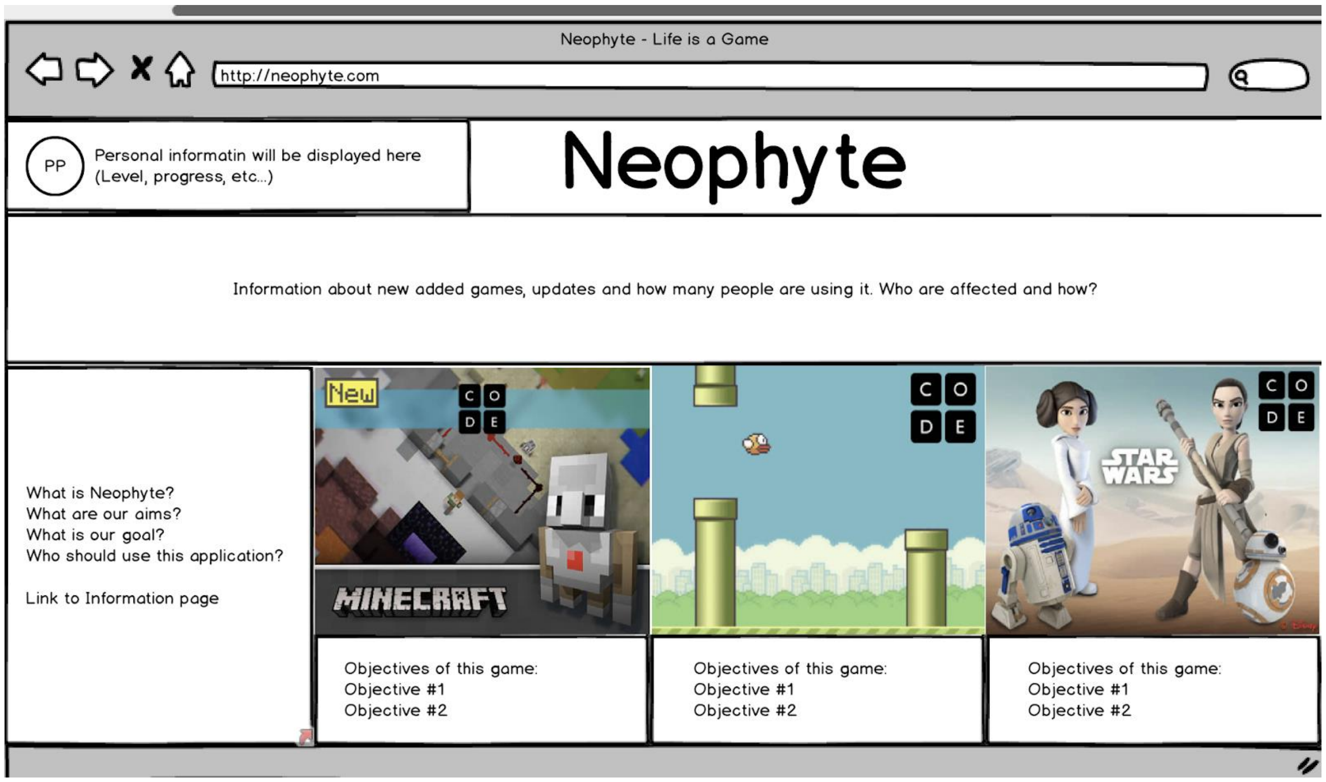


Figure 10: Profile Screen

This page contains all the information about the users progress and goals. There is an informaiton card that shows users's credentials like username, e-mail address, level and golas. There is a window for the recent incomplete game that user played. Another window is for showing the progress the user made and milestones the user reached.

There are three other main info windows in this page. First of them is the current progress window. This window shows which games did the user played and completed and what earnings did the user had. Second one is the certificate window. It shows which certificates the user got and what milestones the user reached in detail. The last one is the section window, if a section exists. This window is a future plan, that if the user is registered under a school, than there will be a section for this user and there will be ranking among users in the system. Those information will be shown here.

Main Page:


Neophyte - Life is a Game

http://neophyte.com

PP

Personal informatin will be displayed here
(Level, progress, etc...)

Neophyte

Information about new added games, updates and how many people are using it. Who are affected and how?

What is Neophyte?
What are our aims?
What is our goal?
Who should use this application?

Link to Information page

Objectives of this game:
Objective #1
Objective #2

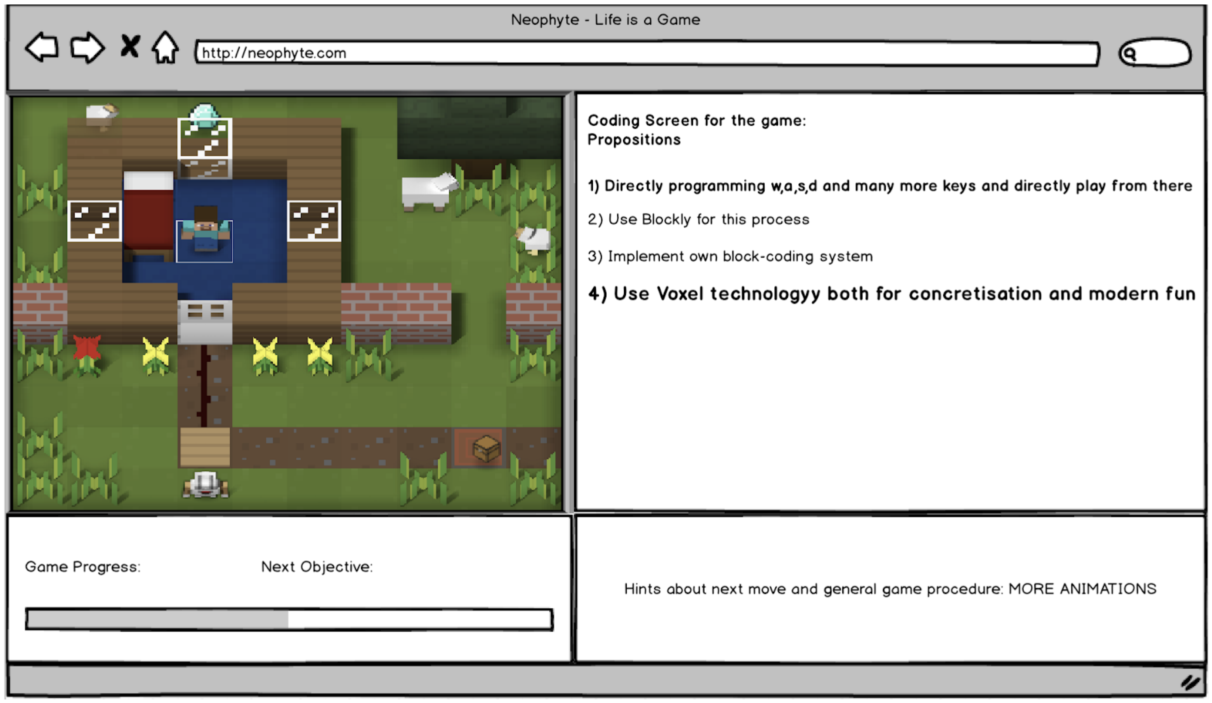
Objectives of this game:
Objective #1
Objective #2

Objectives of this game:
Objective #1
Objective #2

Figure 11: Main Screen

Main page will hold most of the information in single window. This page holds links to the current games, a basic personal information area, a rectangular info area about recent updates, newest game, user comments and many more news about the system; a link to the information page of the application and it also holds the logo of the application on the upper left of the screen.

Game Example #1:



Coding Screen for the game:
Propositions

1) Directly programming w,a,s,d and many more keys and directly play from there

2) Use Blockly for this process

3) Implement own block-coding system

4) Use Voxel technologyy both for concretisation and modern fun

Game Progress: Next Objective:

Hints about next move and general game procedure: MORE ANIMATIONS

Figure 12: Game Screen - 1

This is an example view of a sample game on the site. This page contains 4 blocks of information. First one is the game view. Game view may from 2 perspectives, one of them is third person view (linear) and the other is bird’s eye view. These view options will help children enhance their 3 dimensional modeling skills. The other window is coding window. The dynamics of this window is not fully decided yet because there are many systems for this job and it will take a few more weeks for us to choose between them or to create our own system. Third block is information block about the game, telling the student what is the next mission or how much of the game is complete. Fourth and the last block contain hints about the game. This window holds so much importance about this project since children at ages 9-12 cannot remember all those complicated information at once. So we will teach them about the ways of coding mainly in that block.

This example game is inspired from Code.org's Minecraft game series from Hour of Code section. This game mainly teaches children in a task based approach. There are simple and easy tasks at the beginning and then harder and followed by more complicated tasks one by one.

This approach is good for children since children enjoy completing things in short times and they should not get bored with long assignments.

<u>Game Example #2:</u>

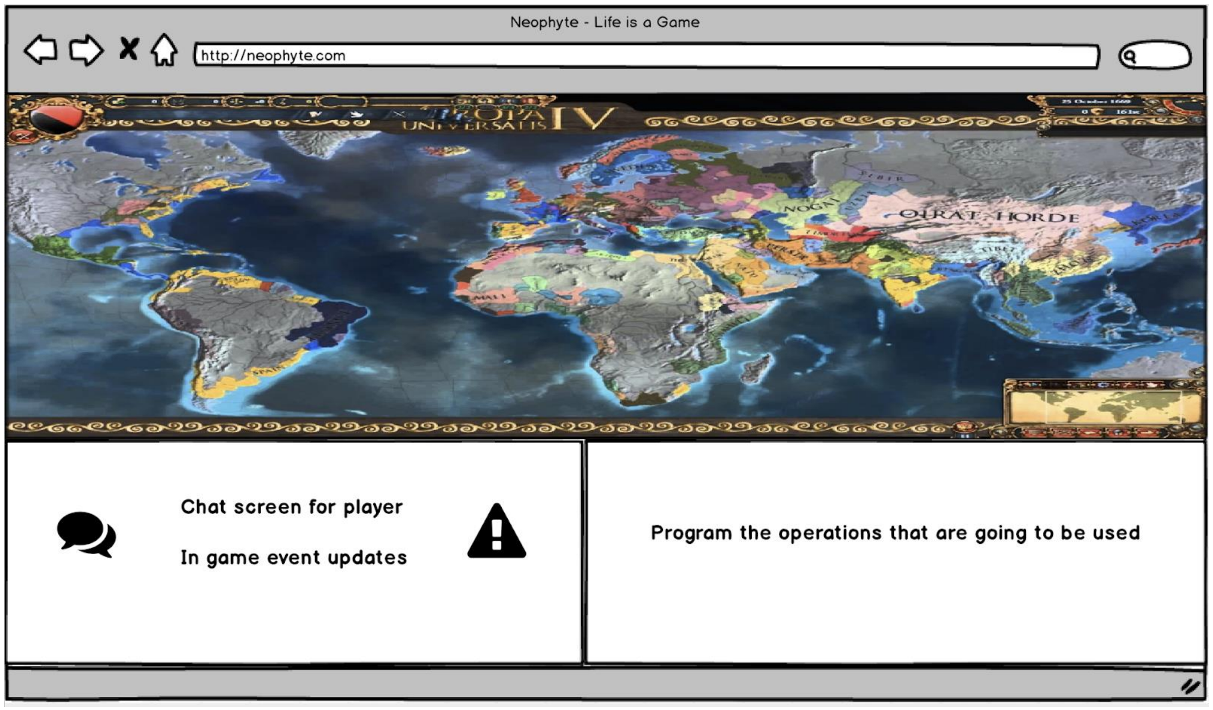

Neophyte - Life is a Game
http://neophyte.com
Chat screen for player
In game event updates
Program the operations that are going to be used

Figure 13: Game Screen - 2

This game is a bit more advanced since it requires strategy skills and offers multiplayer option. This game is more focused on forcing kids to use conditionals and loops. Using these structures in the game will enhance their decision making abilities and at the same time, their coding abilities.

As this game offers a multiplayer option, there will be 3 blocks of information on the screen. First of all, this is the game view window. This window is larger than the other games window because a player also needs to see other players' movements on the map too. More detailed game view will be easier for children to understand what is going on in the game easier. Second block contains the coding options for different operations in the game. This window will be smaller than the other games' because there will be less but more focused operations. Third and the last window is chat and game update window. This window will hold updates in the current game (moves that other players make) and game chat for cooperated games.

This example is not inspired from anywhere. We have been meeting and talking with pedagogues for the last 3 weeks and been trying to create a new system for kids to learn programming in the easiest way. We also have some other game types which will be discussed in following paragraphs.

Application Information Page:

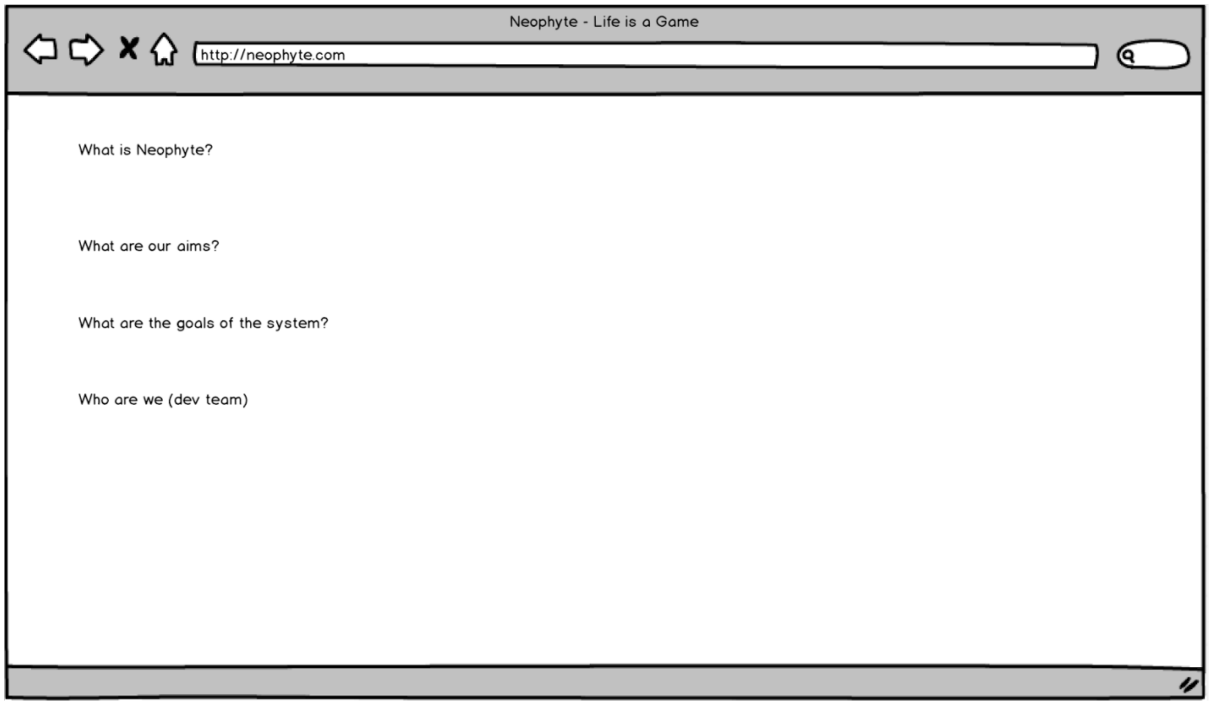


Figure 14: Application Information Page

This page holds the information about the Project Neophyte such as the developers, contact addresses as well as the aim of this project. This page can be accessed from main page by clicking the box named “What is Neophyte?”.

4.0 References

[1] “Coding at school: a parent's guide to England's new computing curriculum,” https://www.theguardian.com/technology/2014/sep/04/coding-school-computing-children-programming Accessed February 17, 2018.

[2] “Adding Coding to the Curriculum,” https://www.nytimes.com/2014/03/24/ world/europe/ adding-coding-to-the-curriculum.html Accessed February 17, 2018.

[3] “İlkokuldan itibaren Kodlama dersi geliyor!,” http://www.sozcu.com.tr/egitim/ ilkokuldan-itibaren-kodlama-dersi-geliyor.html Accessed February 17, 2018.

[4] “Blockly | Google Developers,” https://developers.google.com/blockly/ Accessed February 17, 2018.

[5] “Scratch - Imagine, Program, Share,” https://scratch.mit.edu/ Accessed February 17, 2018.

[6] “Coding Games and Programming Challenges to Code Better,” www.codingame.com/start Accessed February 17, 2018.