Bilkent University

Department of Computer Engineering

Senior Design Project

Project short-name: Neophyte

High Level Design Report

Ali Soyaslan, Oğuz Liv, Umut Akös, Gülce Karaçal

Supervisor: Halil Altay Güvenir

Jury Members: Uğur Güdükbay and Hamdi Dibeklioğlu

Progress Report

May 14, 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 overview of the architecture and design of the system that 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 design are listed. Subsystem decomposition, architectural plans of subsystems, and hardware/software

mapping of these components are illustrated. Design decisions such as persistent data management, access control and security, boundary conditions are reported in detail. Finally, the functions of subsystem services and their interactions are outlined.

1.1 Purpose of the System

The main purpose of Neophyte is to provide a unique platform in which children can learn basic concepts of programming while playing various types of games designed for their entertainment. We aim to raise awareness among children about computer science topics by dragging their attention to these topics with games and fancy graphics.

In order to keep children focused on game, we will need to provide some interesting features about our program. To do so, we will offer an exciting scenario which will be about game flow. Besides the scenario, there will be in-game activities that allow users to proceed next stages. Also, in-game interaction with other users and cooperative missions are what differentiates our program from others because Neophyte will provide a competitive environment for children in which they can easily interact with each other.

1.2 Design Goals

1.2.1 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.

1.2.2 Efficiency

- The system should react to user's input under 10 seconds.
- Load time should be minimal.

- Controls should contain the minimum delay possible.
- The compilation time should be at most 20 seconds.

1.2.3 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.

1.2.4 Security

- The system should ensure security of users' personal data & privacy and in order to address this goal, the application will be accessible only when user logs in to his or her account with his and her password.

1.2.5 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.

1.3 Definitions, acronyms, and abbreviations

UI: User Interface
API: Application Programming Interface
Server: The part of the system which is responsible of logical operations, scheduling and data management
Client: The part of the system that the user interacts
HTTP: Application Layer Protocol

1.4 Overview

Project Neophyte is a learning tool for children in elementary school and middle school ages. Strictly speaking, ages ten to fourteen. 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 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 create a real life simulation that will have real life problems that kids may have. For example, kids might lose their keys or toys, or even their tablets or phones. There will be a quest for children to play in single player mode or multiplayer mode. Throughout the quest, there will be mini-assignments for them to solve. In this quest, they may have to send a message to their friend for him to come down to, for instance, basketball court to play with them. If they want to send this message, they have to get the charge cable from their parents first, then they have to find their tablet in their messed up room by cleaning it up. While cleaning their room, they have to grab their belongings that are scattered around in closest element first fashion.

For kids in that age, it might be a little hard to figure out the code at once, twice or even ever. That's why we will implement an acting game mode for the first levels. They will move their character around the room with arrow keys or W,A,S,D keys and pick up items with space key. This will teach them how to use keyboard as well, since we want to raise kids according to needs of the day. This type of games will get them to recognize the algorithms they will encounter in their education or professional life. Then the games will become more abstract and second levels will contain a box view for coding. It will still hold the concreteness by having blocks all around. Children can play with these boxes in a sandbox fashion. They will create movements and

figures by changing block attributes and arrangements. The final levels will be completely abstract and will contain coding fields for children to finally start real coding.

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 pedologists 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.

2.0 Current Software Architecture

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 in-game trophies based on completing time, lines of codes of the given tasks, which can be decided by test cases. 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 Software Architecture

3.1 Overview

In the subsystem decomposition, the subsystem structure of our system is described thoroughly. Partitions of the system with classes in each layer are shown. Then, system's mapping of hardware/software will be provided which presents different parts of our system and how it works using different hardware components. Persistent data management is also present that explains how we store our data. Access control and security defines access and exit boundaries of our systems. In global software control, how our server acts as main controller is discussed as well as the general flow in the system. In the end, boundary conditions such as initialization, termination and failure conditions.

3.2 Subsystem Decomposition

Neophyte follows a Client-Server architectural style in order to effectively respond/process concurrent user requests. The web application will constitute the client part of the system. The client requests services from the server to function and to respond the needs of the users.

On the server side, the system will be managing database and compiler. This server must be as efficient as possible, because, it processes too much data when command is given by the user. The primary goal is to achieve highest performance while sustaining the lowest response time for each user request. In other words, the client will be focusing on taking the requests of user and on responding them while server will be handling data retrieval tasks and compiler compiles if necessary.

Our system will be divided into 3-Tier system architecture which are: Presentation Tier, Logic Tier and Data Tier. The Presentation Tier, which is the topmost tier, includes the visual components of the system and it can be found mostly on our client-side application. This tier is responsible for managing the interactions with the user. The Logic Tier contains the fundamental operations and all the functionalities of the Neophyte in the Server side of system. Moreover, the Data Tier is responsible for database management and primarily resides on the server side.

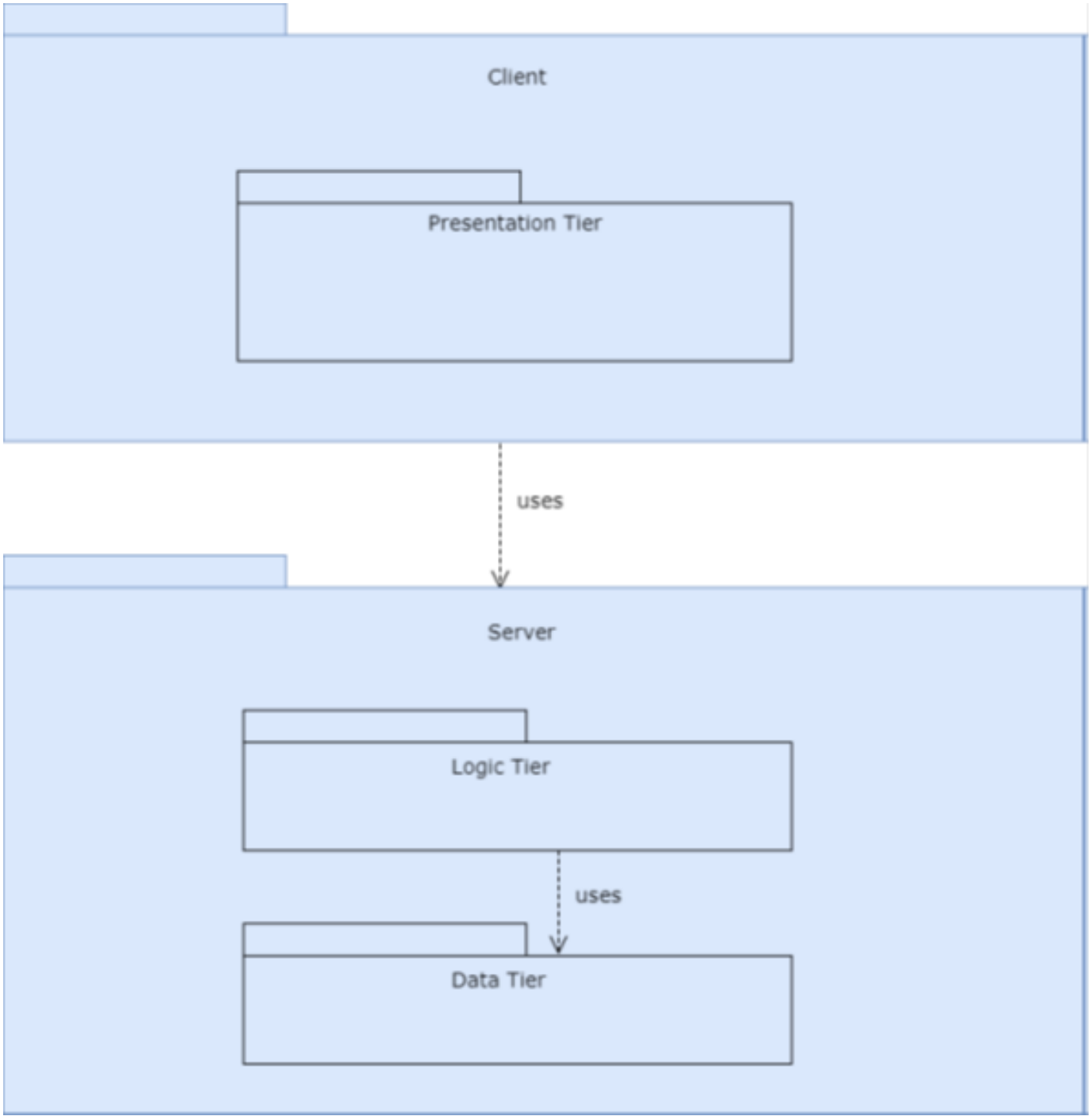

Figure 1: Subsystem Decomposition

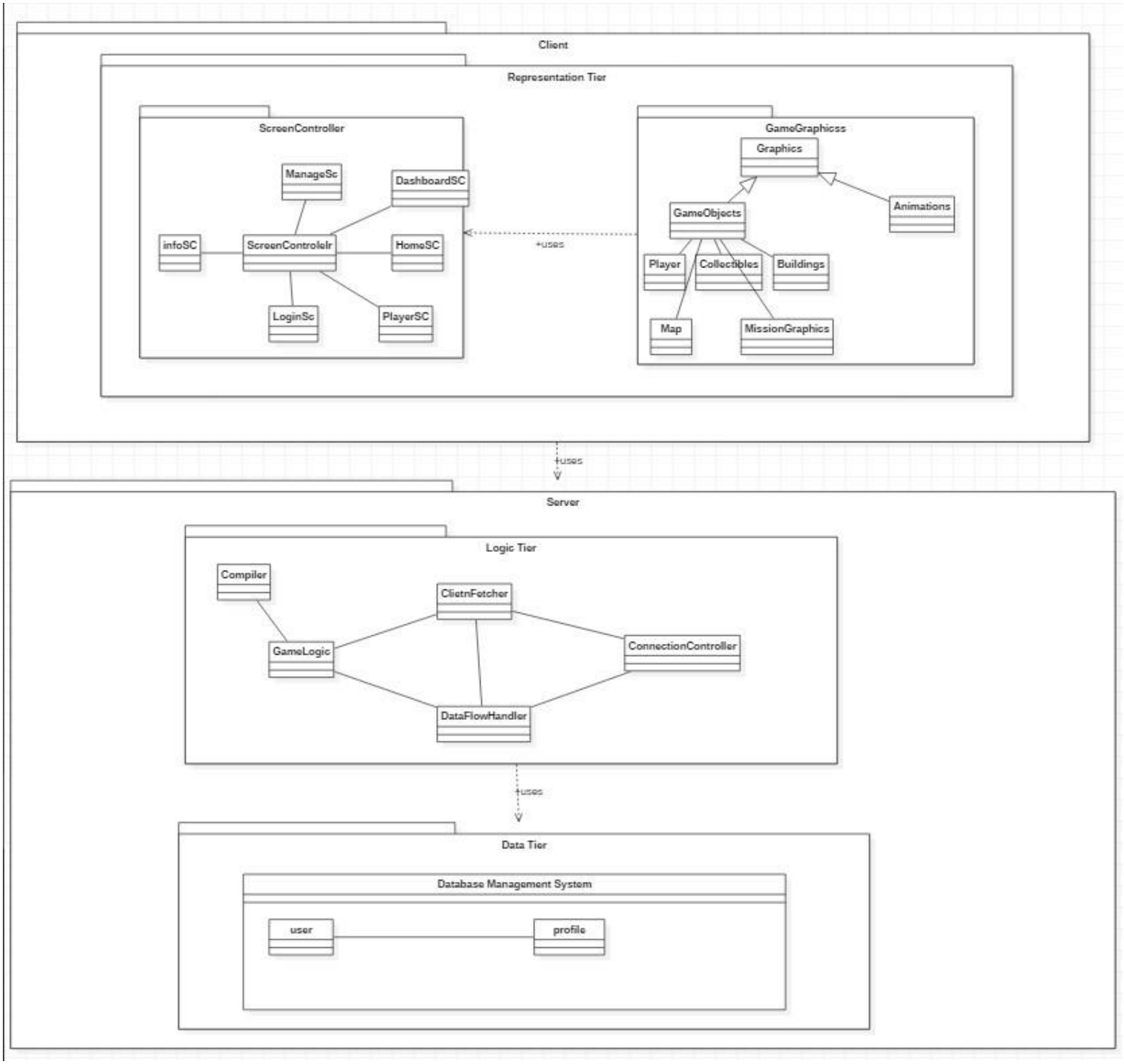

Figure 2: Detailed View of Subsystem Decomposition

3.3 Hardware/Software Mapping

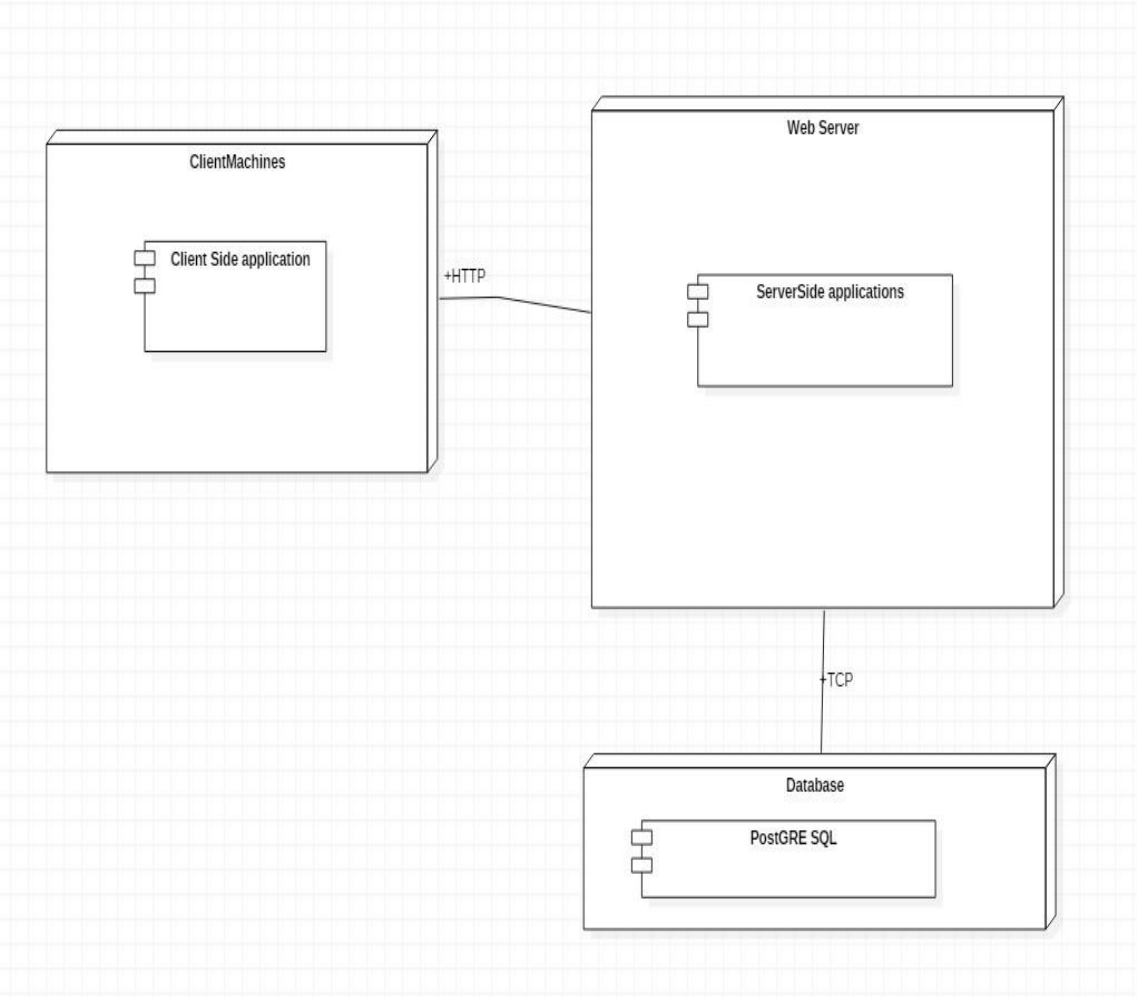

Figure 3: Component Diagram

The client of our application will be any browser on market. As our application will be a web application that will run on a mobile application, we will implement a browser view in a mobile application for tablets. This way, children at school can also use this project from the PC labs in their school. Neophyte will make use of tablet screen to interact with the user. Server side of the system consists of Database Management System. Also, due to the fact that the data will be saved in the Data Server, it will not have significant memory requirement. Internet connection will be necessary for the client machine to retrieve persistent and real-time data from the server side. The Client Machine will communicate with the Web Server using HTTP requests.

3.4 Persistent Data Management

According to our plan, we are going to store various data about users such as name, e-mail address, password, location, company and school. Some of the data that we will store may be changed by users because they might want to change their locations and personal data. Moreover, game scores will be changed frequently as the user completes a game or is beaten by another user during the competition. Also, we might have to response each modification very quickly. In that case, we cannot afford expensive database operations. Hence, we decided to use PostGRE.

3.5 Access Control and Security

We will ask users to sign up and login to the program in order to use the application. Users will be allowed to change their personal information by modifying personal settings. Also, users are able to search other users and to add others as friends in order to play a game together.

Moreover, security is one of the key concerns of Neophyte. All users register to the system with a username/email and a password. The user information will only be shown to the user only. These data will be secured by using third party security system. Users must login in order to access their personal details. Apart from user request, we will not distribute irrelevant user information to third party applications. Furthermore, our application will not access or connect any other application without user's consent.

3.6 Global Software Control

Neophyte will have a centralized event driven control system. The user logins to the system by entering his/her username and password. Then, the system checks whether the username/password combination exists in the database or not. If the combination exists, the application gives the user permission to login to the application. Additionally, in our system, users are able to modify their personal information and to check their scores. When one of these information is changed, Neophyte updates the related user's profile.

3.7 Boundary Conditions

Initialization

After the user launches the app, they will be brought to the login page, where they will be able to enter their credentials and if successful, will be redirected to the main page. Otherwise they will get an error.

Termination

The user can terminate the application by logging off. If he or she decides not to log out, application will keep the account information and keep the user logged in. Apart from logging off, user can also terminate the session by clearing the application data. If the application is not closed, it will run on the background.

Failure

The application can cause a failure if there is no internet connection. Additionally, termination of the application while the application is performing an action, might cause failures.

4.0 Subsystem Services

This part of the report analyzes the subsystems of our system and describes the services they provide in detail.

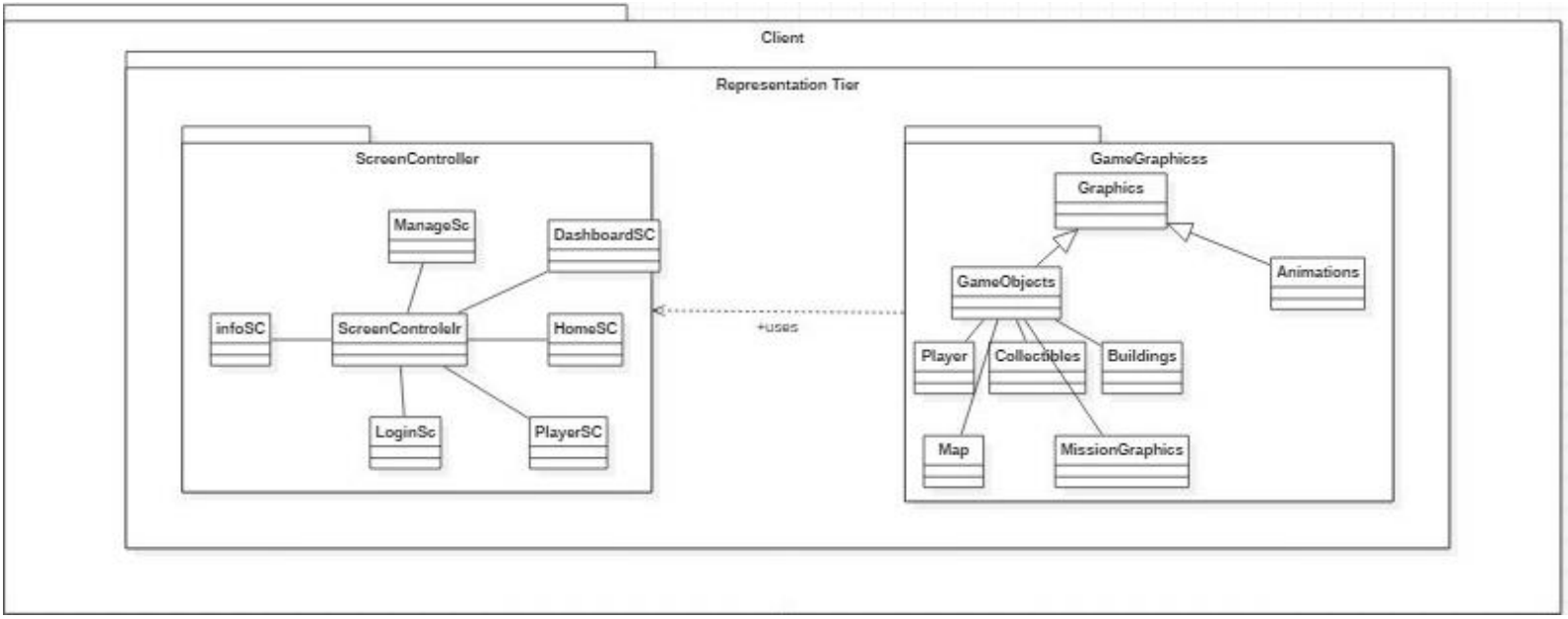

Figure 4: Detailed View of Client Subsystem

4.1 Client

4.1.1 ScreenController Subsystem

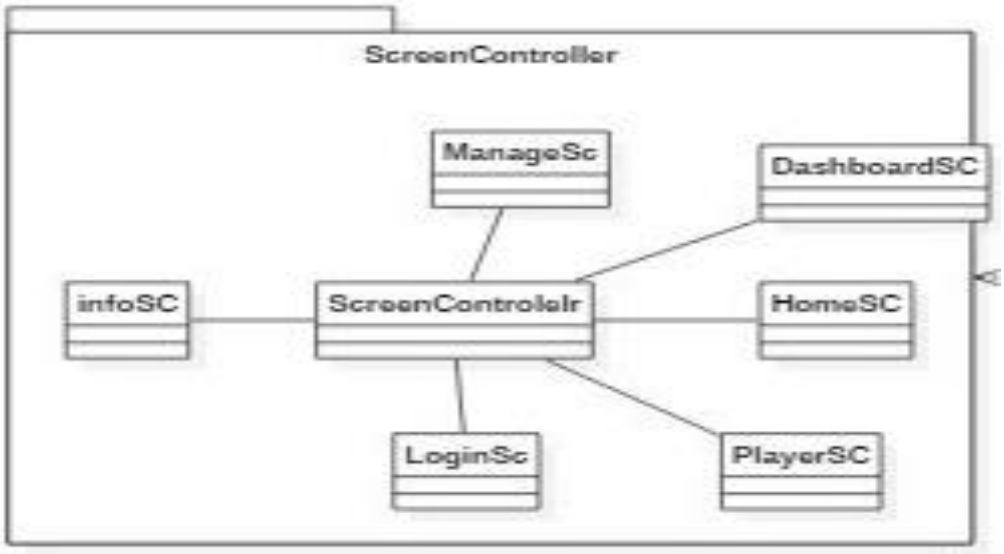

Figure 5: ScreenController Subsystem in Client

- **ScreenController :** It **c**ontrols which screen to be loaded and also handles loading process.
- **LoginSC**: It loads login screen when login button is clicked on home screen.
- **PlayerSC:** It loads the player information such as his/her points, rank and collectibles.
- **HomeSC**: It is the initial opened screen of our program. It offers player to login or get some information about program and its creators.
- **InfoSC:** It loads credits of the game on the screen.
- **DashboardSC**: It loads rankings of the all players in our program. Also, It allows to view player screen when any player clicked on the list.
- **GameSc:** It loads the game when New Game button or Continue button clicked.
- **MessageSC:** It loads message screen when message button clicked on player screen for a specific player.

4.1.2 Graphics Subsystem

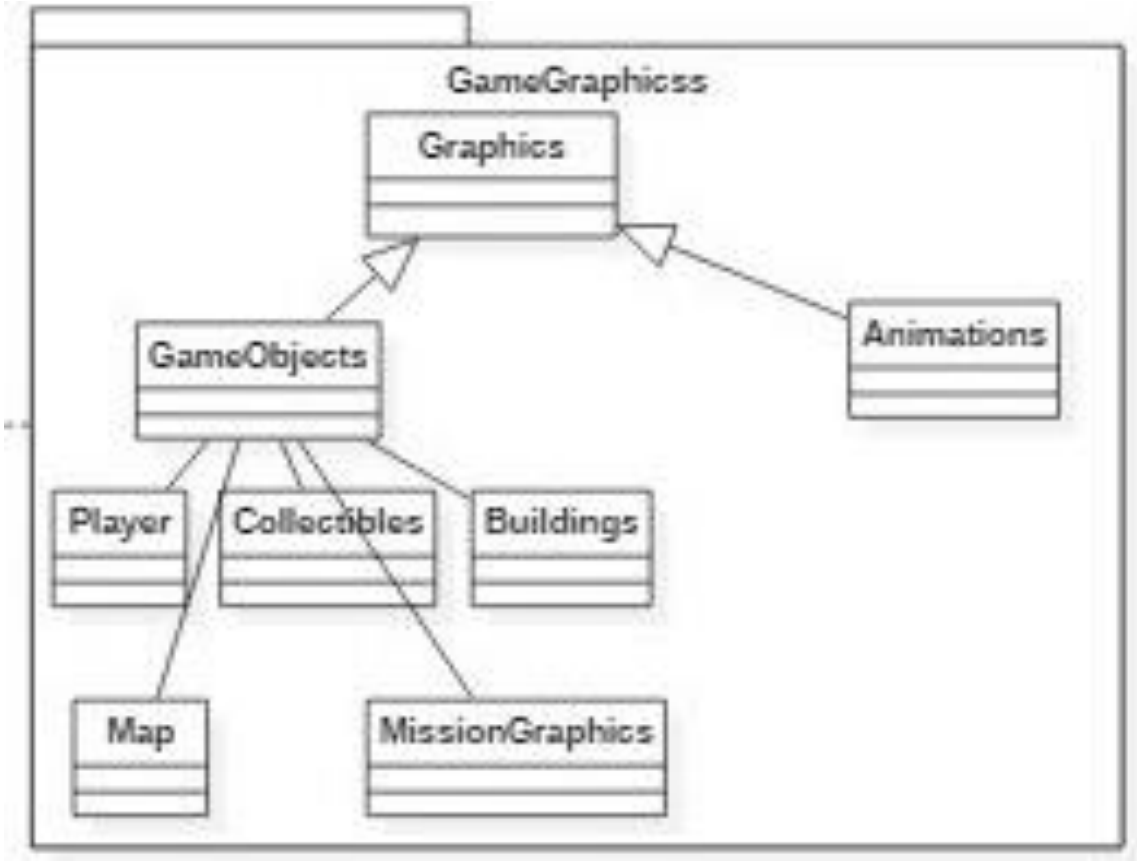

Figure 6: Graphics Subsystem in Client

- **Graphics:** It is the most fundamental attribute for graphics. Other specific graphic objects are inherited from this class and also its functionalities.
- **Player:** It loads player graphics. It depends on the user, so proper graphics are loaded from DB according to player.
- **Collectibles:** It includes the representations of collectibles in the game. Those are cannot be changed so they are also loaded from database according to game flow.
- **Buildings:** Another crucial graphical content of our game is buildings. There are not much building species; however their designs are controlled by this class.
- **MissionGraphics:** According to ongoing mission, graphics are loaded from this class.
- **Map:** It loads all contents of the Map according to selected game mode, game stage and player.

4.2 Server

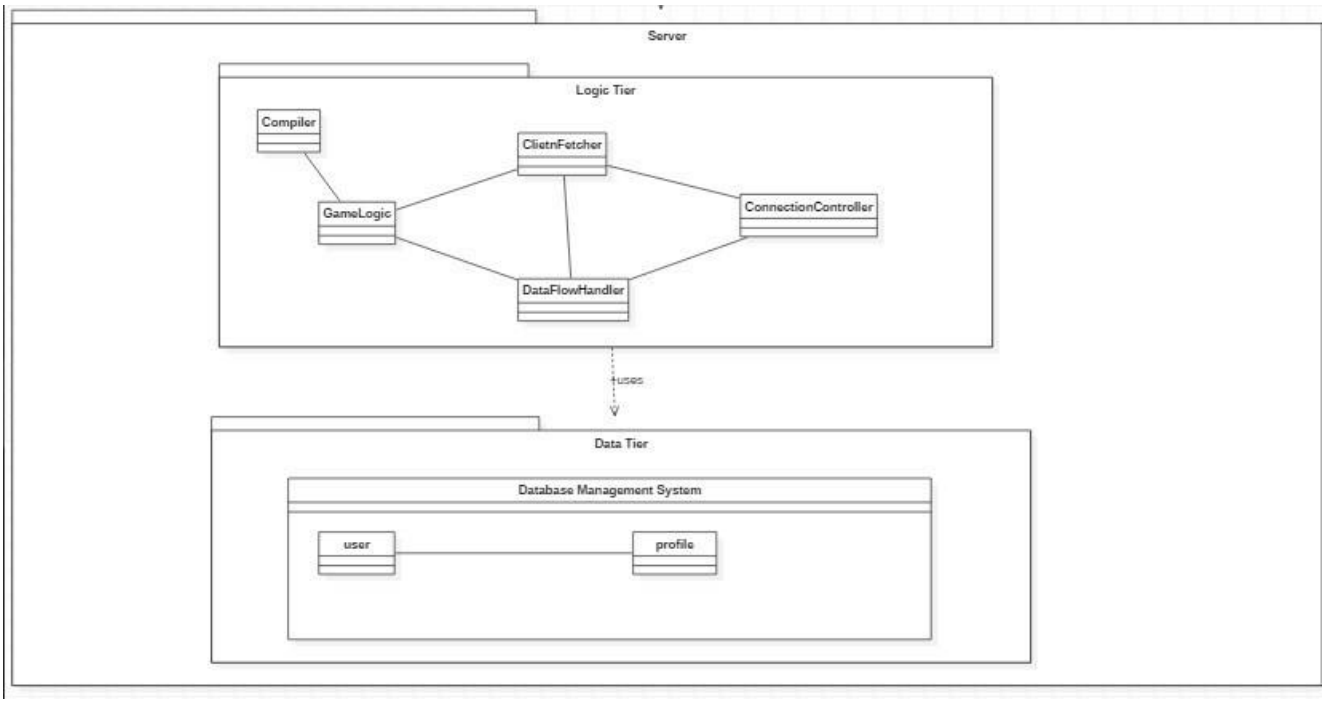

Figure 7: Detailed View of Server Subsystem

Server side is one of the hearts of our project and allows our program to connect to players and its database. Also, in our server architecture we have our compiler in order to compile in game codes. This compiler is going to test in Java.

In our database, all of the entities about players are going to be stored. Also graphics are going to be stored in our database. To proper use of our database, PostGRE SQL will be used. All of the server side components will be deployed from Amazon Web Service (AWS).

Client Side is going to interact with API directly. In order to interact, API is going to fetch instructions from client, and loads proper graphics and content of the program according to request. This process guarantees that there are no direct interactions with data storage. Furthermore, according to game stage, API directs flow to compiler and then continues to control flow according to user commands.

4.2.1 Logic Tier

Logic Tier is the layer which communicates with the database and client. It processes the incoming data. This tier contains Client Fetcher and Data Flow Handler, GameLogic and the ConnectionController, which deal with the core business logic.

- **ClientFetcher:** This class communicates with the client. According to client's commands, this class directs data to Data Flow Handler.
- **Data Flow Handler:** According to command and its perception in the program, this class directs data to proper classes.
- **Game Logic:** This class controls game commands according to user commands. This class directly communicates with client in order to execute client's commands immediately.
- **Connection Controller:** This class handles connection according to game mode. If single player mode is selected, then the server establishes the connection for the single player. If it is selected as multiplayer, then the connection will be established for multiplayer mode.

- **Compiler:** This class compiles codes and returns the result to the data flow handler, if the code is working or not. This class can be used only in programming assignments of the game, that's why it is connected to Game Logic class.

4.2.2 Data Tier

Data tier is the layer which manages the data of software. It consists of only Database Management System. Database Management system handles relational database operations.

- **User:** It contains user's data to be represented in the software.
- **Profile**: It includes in-game user data to be represented according to user in the game.

5.0 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.