Executive Summary

For the final project in MIS 507, our team wrote Java code and used four software design patterns to develop a game system specifically for elementary schools. This game system allows elementary students (players) to play either a typing game or a mathematics game. In the typing game, the student can practice by typing the home row keys or by typing on all of the keys to create general paragraphs. In the mathematics game, the student can practice addition, subtraction, multiplication, and division problems. After the student (player) completes a game, their score is saved along with their profile. The next time the student (player) plays the game (typing or mathematics) they compete against themselves to try to beat their previous high score for that game. If their score is higher, they win and get a new high score. The overall objective of this game system is to help elementary students improve their typing and math skills by playing the games we created. A high level outline of the entire document is shown below.

Section 1	Provides an explanation about why and how the game system was developed specially for Elementary Schools.
Section 2	Contains the UML diagram for the game system and explains why the Observer, Strategy, Template, and Singleton design patterns were selected.
Section 3	Explains a specific use case: a student playing the typing game and then beating their previous high score for the game. The environmental details are explained as the typing game is played.
Section 4	Provides a summary of the lessons learned and the specific takeaways that our team gained from the process of designing and writing the code necessary to complete the game system.

Table 1: Team members and their role on the team.

Name	Role on Team	Email
Kory Chinn	Final Report, Game Coding and Testing	knc1@email.arizona.edu
Karan Dhamija	Business Logic creation and Demo Video Creation	karandhamija@email.arizona.edu
Po-Yi Du	Coding, Business Logic creation, Testing, and UML Diagram creation	pydu@email.arizona.edu
Derek Gourley	Final Report, Game Coding and Testing	gourleyd@email.arizona.edu
Yazan Abu Hijleh	Coding, Business Logic creation, Testing, and UML Diagram creation	yna@email.arizona.edu
Tso-Ming Liu	Coding and Demo Video creation	tsomingliu@email.arizona.edu

Section 1 - Introduction

The game system that we are developing was designed to fit in with the current elementary school system and this served as the business scenario that our team had in mind as we created the business logic and wrote the Java code.

As technology becomes more prevalent in elementary schools within the United States, our team wanted to develop a game system that could be integrated with the current curriculum, while being flexible enough to adapt to the ever frequent changes to the curriculum that occur from year to year. In addition, the source code for the game system was written in such a way that it could easily be adapted to meet the specific curriculum requirements for each school.

The need that this game system will fill, is that it will provide an easy way for teachers to teach math and typing skills, all while making it fun and engaging for students. In talking with elementary school teachers, we continuously heard that it is difficult to keep the attention of the students for long periods of time. We also heard that teachers were having difficulty adjusting to the push for technology to be incorporated in everyday use in the classroom. After hearing what the elementary school teachers had to say, we set out to create a game system that would meet the needs of the teachers and the curriculum as well as to try to keep the attention of the students.

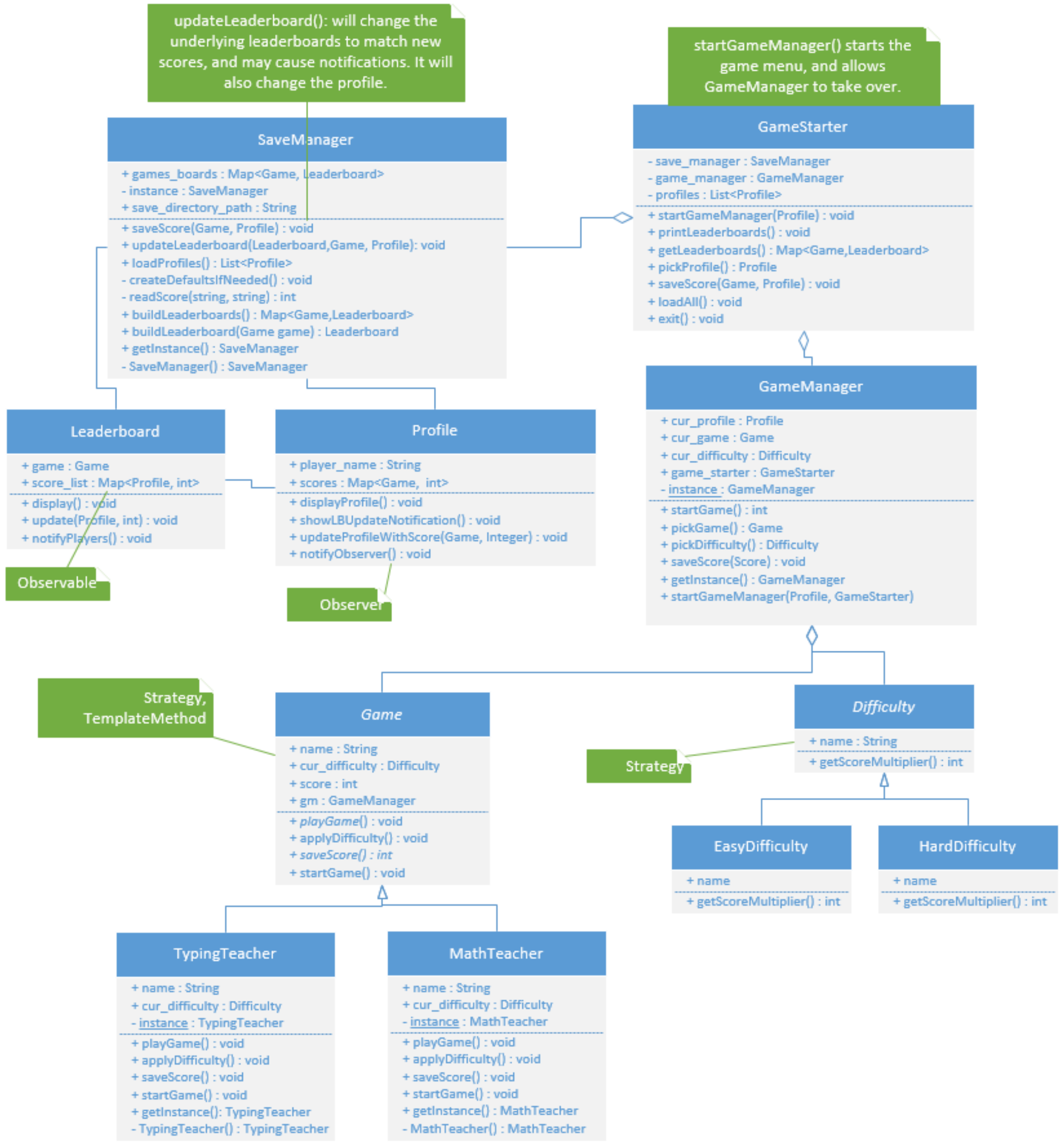
After our team had identified, researched, and explored the target market and business scenario that we would be developing our game system for, we began exploring software design patterns. School systems have constantly evolving requirements for teaching and our team was specifically looking for software design patterns that could be used in a way to make it easy to make changes to the games themselves as the curriculum changed. In addition, it is important to design the game system in a way that the score of each student (player) can have their scores recorded and saved for the end of the year where teachers need to assess and document the progress that each student made during the year.

The benefit of this game system would be that it would help teachers to follow and track the progress of each student in regards to their typing and mathematics skills as the school year progresses. In addition, it would provide a leaderboard that allows the teacher to better understand which students (players) are doing well and also show those students who might need more practice or special instruction to improve either their typing or mathematics skills. The end result is a game system that helps teachers teach typing and mathematics, all while recording each students (players) scores and making it fun and entertaining for students.

Section 2 - System Design

The UML diagram that we created for our game system is shown in figure 1 below. Our team first created the UML diagram and then referenced it to write the Java skeleton code, which we then filled in to finalize the game system.

Figure 1: UML diagram for the game system.


updateLeaderboard(): will change the underlying leaderboards to match new scores, and may cause notifications. It will also change the profile.
startGameManager() starts the game menu, and allows GameManager to take over.
SaveManager
+ games_boards : Map<Game, Leaderboard>
- instance : SaveManager
+ save_directory_path : String
+ saveScore(Game, Profile) : void
+ updateLeaderboard(Leaderboard,Game, Profile): void
+ loadProfiles() : List<Profile>
- createDefaultsIfNeeded() : void
- readScore(string, string) : int
+ buildLeaderboards() : Map<Game,Leaderboard>
+ buildLeaderboard(Game game) : Leaderboard
+ getInstance() : SaveManager
- SaveManager() : SaveManager
GameStarter
- save_manager : SaveManager
- game_manager : GameManager
- profiles : List<Profile>
+ startGameManager(Profile) : void
+ printLeaderboards() : void
+ getLeaderboards() : Map<Game,Leaderboard>
+ pickProfile() : Profile
+ saveScore(Game, Profile) : void
+ loadAll() : void
+ exit() : void
GameManager
+ cur_profile : Profile
+ cur_game : Game
+ cur_difficulty : Difficulty
+ game_starter : GameStarter
- instance : GameManager
+ startGame() : int
+ pickGame() : Game
+ pickDifficulty() : Difficulty
+ saveScore(Score) : void
+ getInstance() : GameManager
+ startGameManager(Profile, GameStarter)
Leaderboard
+ game : Game
+ score_list : Map<Profile, int>
+ display() : void
+ update(Profile, int) : void
+ notifyPlayers() : void
Observable
Profile
+ player_name : String
+ scores : Map<Game, int>
+ displayProfile() : void
+ showLBUpdateNotification() : void
+ updateProfileWithScore(Game, Integer) : void
+ notifyObserver() : void
Observer
Strategy, TemplateMethod
Game
+ name : String
+ cur_difficulty : Difficulty
+ score : int
+ gm : GameManager
+ playGame() : void
+ applyDifficulty() : void
+ saveScore() : int
+ startGame() : void
Strategy
Difficulty
+ name : String
+ getScoreMultiplier() : int
EasyDifficulty
+ name
+ getScoreMultiplier() : int
HardDifficulty
+ name
+ getScoreMultiplier() : int
TypingTeacher
+ name : String
+ cur_difficulty : Difficulty
- instance : TypingTeacher
+ playGame() : void
+ applyDifficulty() : void
+ saveScore() : void
+ startGame() : void
+ getInstance(): TypingTeacher
- TypingTeacher() : TypingTeacher
MathTeacher
+ name : String
+ cur_difficulty : Difficulty
- instance : MathTeacher
+ playGame() : void
+ applyDifficulty() : void
+ saveScore() : void
+ startGame() : void
+ getInstance() : MathTeacher
- MathTeacher() : MathTeacher

The game system was developed with the primary use case being to help teachers in elementary schools in the United States teach typing and mathematics to students. In the UML diagram shown in figure 1 above, the game system utilizes four software design patterns: Observer, Strategy, Template, and Singleton.

Design Patterns:
The observer software design pattern was used in the game system to handle any changes to the leaderboard. If the player's score is better than their previous score, the Observer software design pattern is responsible for updating the score for each player. The observer in this case is the players profile and the observable is the leaderboard.

The strategy software design pattern was used within the game system to make changes to the difficulty of the game and to switch between the games (currently two). In addition, the strategy design pattern was used to allow for new games to quickly be added or removed without requiring any major code revisions. We expect that with the often rapid changes to each elementary schools curriculum, that the use of this design pattern will be critical to quickly extending and adapting the type of games offered both now and in the future.

The template software design pattern was used in the game system to standardize the process of setting up a game: 1) Apply the difficulty, 2) Play the game, and 3) Return the players score. The usage of the template software design pattern is helpful now even with two games, but will also become increasing more useful in the future when additional games are added in order to adapt to changes in each elementary schools curriculum.

The singleton software design pattern was used in the game system to ensure that only a single instance of both the game manager and the save manager classes exist as well as to ensure that only a single game is created at a time. In addition, the usage of the singleton software design pattern will help to protect the integrity of the files and objects used by the game system.

Key Features:
Each of the design patterns used in the game system were selected to make the entire code base follow the "Open-Closed" principle. The goal of this principle is to write code that is open for extension and closed for modification. Given that our team developed a game system specifically for elementary schools, an environment where the teaching curriculum is rapidly changing, this principle will be especially important in allowing for quick changes to be made to the game system in order to prevent the games from becoming obsolete.

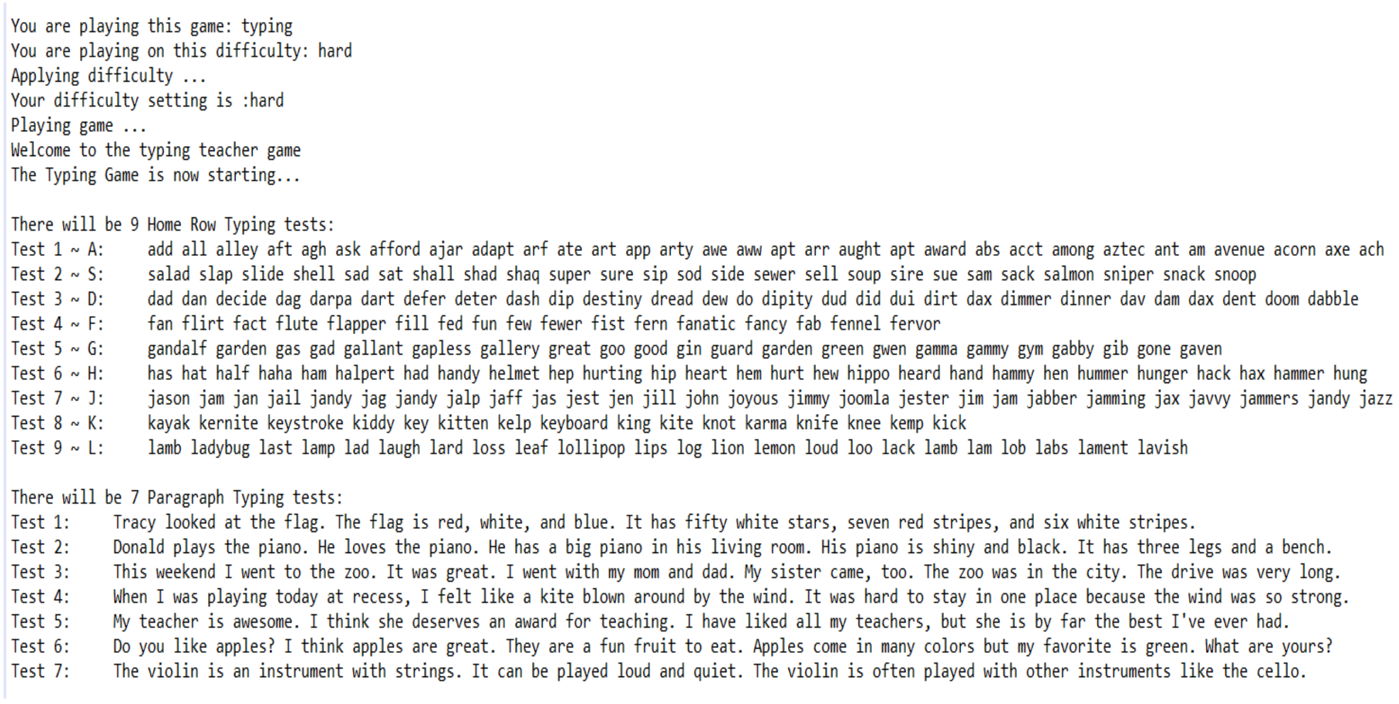
The game system currently has two games: typing and mathematics, but the entire system was designed to be extensible so that each individual school could edit or add additional games that are tailored to specifically meet the requirements of their curriculum.

Section 3 - System Implementation
Implementation of our game software was completed using an object-oriented design in Java by utilizing the design patterns described in Section 2. The code was split amongst team members and each of the team members were responsible for the implementation of 2 or more classes in our UML diagram. The team used GIT for versioning control and

created branches from our Bitbucket repository to develop our individual sections. These features were then merged back into our master branch. The actual development was completed using the integrated development environment, Eclipse.

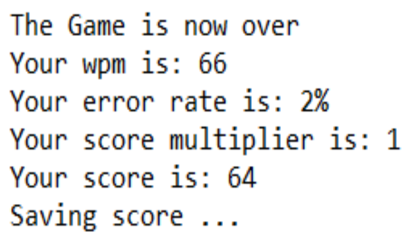
To demo and test the game system, we created a use case where a student named Yazan plays the typing game. Figure 2 below shows Yazan playing the game (on hard difficulty) and Figure 3 below, shows the score he got as well as some in-game metrics related to the typing game Yazan just completed. After Yazan has finished playing the typing game, the game system then checks to see if his scores are better than previous scores that he has gotten, if they are better, then the leaderboard gets updated. If Yazan decided to play the math game, the results would be similar, except instead of typing questions, he would get asked math questions (addition, subtraction, multiplication, and division).

Figure 2: Yazan is playing the typing game.



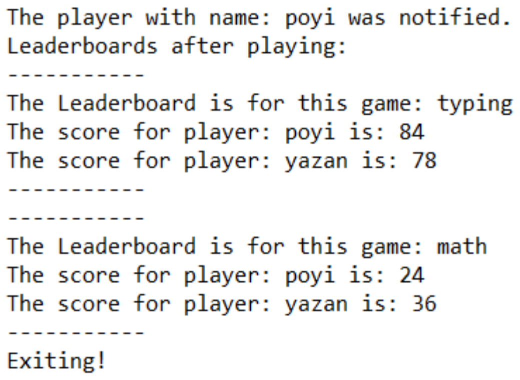
```
You are playing this game: typing
You are playing on this difficulty: hard
Applying difficulty ...
Your difficulty setting is :hard
Playing game ...
Welcome to the typing teacher game
The Typing Game is now starting...

There will be 9 Home Row Typing tests:
Test 1 ~ A:     add all alley aft agh ask afford ajar adapt arf ate art app arty awe aww apt arr aught apt award abs acct among aztec ant am avenue acorn axe ach
Test 2 ~ S:     salad slap slide shell sad sat shall shad shaq super sure sip sod side sewer sell soup sire sue sam sack salmon sniper snack snoop
Test 3 ~ D:     dad dan decide dag darpa dart defer deter dash dip destiny dread dew do dipity dud did dui dirt dax dimmer dinner dav dam dax dent doom dabble
Test 4 ~ F:     fan flirt fact flute flapper fill fed fun few fewer fist fern fanatic fancy fab fennel fervor
Test 5 ~ G:     gandalf garden gas gad gallant gapless gallery great goo good gin guard garden green gwen gamma gammy gym gabby gib gone gaven
Test 6 ~ H:     has hat half haha ham halpert had handy helmet hep hurting hip heart hem hurt hew hippo heard hand hammy hen hummer hunger hack hax hammer hung
Test 7 ~ J:     jason jam jan jail jandy jag jandy jalp jaff jas jest jen jill john joyous jimmy joomla jester jim jam jabber jamming jax javvy jammers jandy jazz
Test 8 ~ K:     kayak kernite keystroke kiddy key kitten kelp keyboard king kite knot karma knife knee kemp kick
Test 9 ~ L:     lamb ladybug last lamp lad laugh lard loss leaf lollipop lips log lion lemon loud loo lack lamb lam lob labs lament lavish

There will be 7 Paragraph Typing tests:
Test 1:     Tracy looked at the flag. The flag is red, white, and blue. It has fifty white stars, seven red stripes, and six white stripes.
Test 2:     Donald plays the piano. He loves the piano. He has a big piano in his living room. His piano is shiny and black. It has three legs and a bench.
Test 3:     This weekend I went to the zoo. It was great. I went with my mom and dad. My sister came, too. The zoo was in the city. The drive was very long.
Test 4:     When I was playing today at recess, I felt like a kite blown around by the wind. It was hard to stay in one place because the wind was so strong.
Test 5:     My teacher is awesome. I think she deserves an award for teaching. I have liked all my teachers, but she is by far the best I've ever had.
Test 6:     Do you like apples? I think apples are great. They are a fun fruit to eat. Apples come in many colors but my favorite is green. What are yours?
Test 7:     The violin is an instrument with strings. It can be played loud and quiet. The violin is often played with other instruments like the cello.
```

Figure 3: Yazan has finished the typing game and gets information about how he played.



```
The Game is now over
Your wpm is: 66
Your error rate is: 2%
Your score multiplier is: 1
Your score is: 64
Saving score ...
```

Figure 5: Leaderboard updating.



```
The player with name: poyi was notified.
Leaderboards after playing:
-----------
The Leaderboard is for this game: typing
The score for player: poyi is: 84
The score for player: yazan is: 78
-----------
-----------
The Leaderboard is for this game: math
The score for player: poyi is: 24
The score for player: yazan is: 36
-----------
Exiting!
```

Section 4 - Lessons Learned

For this project, our group consisted of team members with varying levels of coding experience. From the beginning of the term, our team set up weekly meetings to discuss the project for this class and to also review the materials covered in lecture. Looking back, starting early on this project was critical to developing a UML diagram and skeleton code that was later used as an outline to complete this project. For this project our team used GIT, which was extremely helpful to have for this project and made collaboration very easy. We specifically used Atlassian BitBucket and combined it with the Eclipse Desktop Software, to create a development environment, which made it easy to write code, push changes, and pull new modifications made by other team members.

During the class this term we learned about the common software design patterns and how they can be used to create source code that follows the “Open-Closed” principle, which we also talked about in class. We also learned that software design patterns can be thought of as an advanced form of object oriented programing, the goal of which is to develop source code that is open for extensibility, but closed for modification. By learning these software design patterns, we were able to write the source code for the game system in a way that allows for easy modification in the future, even by programmers other than ourselves. The key takeaway from all of this is the increased level of comfort that all of the team members have when it comes to writing code in a format that could be used in a production environment. Before this class, the majority of the code that our team members wrote were for ourselves and not really designed to be used by other programmers. This project gave us experience using common software design patterns that other programmers will be familiar with and these programmers would be able to understand and recognize the software design patterns used when they saw the source code. The end result is code that can be easily modified and adapted as future requirements change and as the project gets handed down to different programmers as time goes on.

A few other key takeaways from this project that we learned and will use in the future have to do with starting early and using software that allows for real time collaboration either remote or in person. By starting early, our team was able to put together a well thought out software design that resulting in making it easier to write the code. In addition, starting early allowed us to make several revisions and iterations on our initial proposal. We were able to take the time needed to develop a good end product, without needing to rush at the end. Setting up regular meetings is not easy with how busy the MIS program can be, but it is an important takeaway for when we enter the workforce.

Our team also utilized several pieces of software that allowed for real time collaboration, Google Drive, Google Docs, and GIT. All of these tools allowed us to work in parallel, by allowing us to see the changes that any of the other teammates made. This was extremely important when it came to writing the code. Given how busy everyone's schedule was, it was necessary to use tools that would allow for the team members to work on their own schedule and prevent them from doing the same work as another member. These tools allowed for everyone in the team to know exactly what everyone else was doing or had done.

Overall, there were many more takeaways from this course and this project that we learned. In this report we just highlighted a few of the lessons learned that we identified at this time. In the future, when we are working out in the industry, we are sure that there will be many more lessons learned from this course that will become apparent when we are doing this work full time. In summary, our team learned about coding, design patterns, collaborative software, and also how to work as a team, which are all things that will be absolutely necessary to be successful in the industry.