EECE 4520 – Software Engineering

Software Design Specification (SDS)
RFID Arcade

Connal West
Nicholas Sullo
Ben Prisby
Omar Tuffaha

1. Introduction

1.1 Purpose of this Document

The objective of this document is to detail the design of the RFID Arcade that BCON is planning. This design specification will attempt to educate all of the developers how to implement the design and instruct the testers how to verify that the product behaves appropriately. The document will address how a game in the arcade should react to external events, the expected behavior of an RFID card, the many different ways that a kiosk is used, the functionality of the rewards center, and the purpose of the backend server.

1.2 Scope of the Development Project

The RFID Arcade is a series of networked machines communicating with a backend that will facilitate the playing of arcade games, ticket generation, and prize redemption. Arcade owners will be able to view statistics of most arcade games played as well as the most popular days for each arcade game. Players will be issued an RFID card using an available kiosk by exchanging currency for tokens, which are necessary to play arcade games. The RFID card serves as a unique identifier for the player within the system. Players may elect to participate in games which publish statistics to the backend. Additionally, players will be able to redeem their tickets for prizes, which can be selected at a special kiosk. Kiosks will be able to display a player's total number of tokens, tickets available for redemption, and interesting gameplay statistics as customized by the client.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
Arcade Game	Machine offering gameplay, which require a set number of tokens for a round and may reward tickets to the player depending upon the design of the game.
Arcade Staff	People who staff the arcade, responsible for machine maintenance and distributing redeemed prizes.
Backend	Server responsible for managing the database storing player, game, and prize information as well as supporting requests to read, write, update, and delete data.
Kiosk	Machine players use to put tokens on their RFID card, view statistics on games played, and view token and ticket balances.
Player	A customer of the arcade which participates in offered

	services.
Player ID	The unique identifier for a player.
Prize	Products stocked by the arcade that can be redeemed by players in exchange for tickets.
Rewards Center	Special type of kiosk that allows a player to browse the prize catalog and redeem prizes.
RFID Card	Physical key identifying a player traveling throughout the arcade storing a player ID and token balance, used to interact with games and kiosks.
Tickets	Form of reward given to players by supported games used to enable the redemption of prizes at a Rewards Center.
Tokens	Internal arcade currency used to play games, purchased at a kiosk with standard money.

1.4 References

[1] Prisby, Ben. Tuffaha, Omar. Sullo, Nicholas. & West, Connal. “Software Requirements Specification (SRS) RFID Arcade.” BCON. 2/12/2019.

[2] “ETEKJOY ACR122U NFC RFID 13.56MHz Contactless Smart Card Reader Writer w/USB Cable, SDK, 5X Writable IC Card.” *Amazon*, Amazon, www.amazon.com/ETEKJOY-ACR122U-13-56MHz-Contactless-Writable/dp/B07FCLY4S9/ref=pd_ybh_a_1?_encoding=UTF8&psc=1&refRID=9AV7ZY0QPD5MCM4BF7PW.

1.5 Major Software Requirements

- Kiosks shall be able to distribute new RFID cards to players with a new player ID requested from the backend and (if applicable) token balance.
- Kiosks shall be able to add tokens to RFID cards.
- Kiosks shall be able to display last played game, total number of tickets, and total number of tokens to a player.
- Arcade games shall be able to debit the number of tokens on a card by the amount required for a single round of gameplay.
- Arcade games shall be able to publish gameplay statistics to the backend.
- Rewards Centers shall be able to display a browsable prize catalog to the player.
- Rewards Centers shall be able to redeem a prize by debiting a player’s ticket balance by the prize amount and request retrieval of the prize.
- Staff and players shall be able to see visually-friendly statistics such as top scorers and ticket earners.

1.6 Design Constraints, Limitations

1.6.1 Design Constraints

Due to the physically distributed nature of the components deployed throughout the arcade, each must be networked. Thus, a robust internal network is required within the arcade to ensure all components can securely and reliably communicate with one another. Ideally, this network should not have online access and be on a separate VLAN from other arcade components such that an intrusion could be controlled. As the backend serves as the central communication hub for the system, it should also be physically secured to prevent unauthorized access.

1.6.2 Limitations

The team is limited by the resources and time available for this project. Given the April 2019 deadline, the developers must prioritize basic functionality over lower-priority functionality to meet MVP. Due to the time limitation, there will likely not be a lot of quality assurance going into the product. There will be a brief testing period after the project is finished, but development is the major concern of the project. The team is also limited by budget. Due to the project being academic-related, the team will not be able to construct a real arcade with actual arcade games and offer legitimate prizes. Instead, there will be a simulation of an arcade for purposes of demoing the overall architecture.

1.7 Changes to requirements

There are no changes to any requirements at this time.

1.8 Overview of Document

The succeeding sections in this document shall traverse the technical details of the system. The Data Design section will detail the specific data that are tracked within the system by category along with information about how each is represented in the backend. The System Architecture Description section will discuss more thoroughly each component within the system at an integration level. Each of these components will then have its individual implementation discussed in Detailed Description of Components. The Interface Design section will discuss more on the specific user interactions that will occur with each machine in the arcade, building on the previously-discussed technical details. The Software design goals will be outlined next, which corresponds to the non-functional requirements outlined in the SRS. Finally, the Appendices contains various diagrams modeling the system and events that may occur within.

2. Data Design

The backend is the centralized hub for storing data within the system. As such, it defines and enforces schemas to provide structure and context for data passed throughout the system. For speed and flexibility, MongoDB serves as the DBMS managed by the backend. It is scalable and

robust in that clients who may wish to track more or less information will be supported, as adjusting schemas can be done with little code impact.

The data are grouped into the following categories, each of which has an individual schema which governs storage within its collection in the database. Types are one of the choices supported by the MongoDB specification.

2.1 Game Data

Property	Type	Required	Unique	Description
_id	`ObjectId`	Yes	Yes	Generated ID from the database.
name	`String`	Yes	No	Display name within the arcade.
tokenCost	`Number`	Yes	No	Tokens required for a single game.
topPlayer	`ObjectId`	No	Yes	The ID of the top-scoring player.

2.2 Player Data

Property	Type	Required	Unique	Description
_id	`ObjectId`	Yes	Yes	Generated ID from the database.
firstName	`String`	Yes	No	First name of the player.
lastName	`String`	Yes	No	Last name of the player.
screenName	`String`	Yes	Yes	Screen name selected by the player for display within the arcade.
tickets	`Number`	Yes	No	Total ticket balance earned by the player.

gameStats	`Array`	No	No	Gameplay statistics (see below).

2.2.1 Game Statistics

The *gameStats* player property is an array of JSON objects published by an arcade game once a game has finished that contains the following:

Property	Type	Description
gameId	`ObjectId`	ID of the game.
ticketsEarned	`String`	Total tickets earned by the player on this game.
gamesPlayed	`String`	Total number of games played by the player on this game.
highScore	`String`	High score of the player on this game.

There shall only be a single entry per game in this player property. The server will check to see if properties need to be updated once games publish data (i.e. an improved high score).

2.3 Prize Data

Property	Type	Required	Unique	Description
_id	`ObjectId`	Yes	Yes	Generated ID from the database.
name	`String`	Yes	No	Display name within the arcade.
description	`String`	No	No	Brief user-facing prize description.

ticketCost	`Number`	Yes	No	Tickets required for a single unit.
availableQuantity	`Number`	Yes	No	Number of available units.
image	`Object`	No	No	Prize image, consisting of a data buffer and content type.

3. System Architecture Description

3.1 Overview of Modules / Components

The Arcade system consists of 7 components that are interconnected to each other within the local network. The components are classified into 3 subsystems based on functionality that allows for high cohesion with low coupling. At least one instance of each component is required to realize the full functionality of the system, though there would likely be more. The System Architecture Diagram of Appendix F shows the following subsystems.

3.1.1 Playing Subsystem

The first component is the RFID card, which stores the unique player ID and token amount. This card is required to interact with games and kiosks throughout the arcade as a means of reliably identifying the player to each machine. RFID cards can be reused as long as a new player ID is issued to reinitialize it.

The second component is the RFID reader, a device responsible for reading and writing data to RFID cards. These shall be integrated into each game and kiosk to support user identification and (in the case of games) token reduction. The reader features an API that abstracts managing the data fields stored on the card.

The third component is the arcade game itself, which is the primary way the player will spend purchased tickets and collect tickets (if supported by the game). Games can take a wide range of shapes and sizes; this system is designed to support retrofitting onto any game with some small modifications.

Together, these three components represent the core functionality of the system: playing games at the arcade. The typical player workflow represents purchasing tokens and thus receiving an RFID card before proceeding to play any number of games.

3.1.2 Kiosk Subsystem

The fourth component is the Rewards Center, a specific type of kiosk that handles prize redemption functionality. It provides a user interface for browsing available prize inventory as requested from the backend. As it also has an integrated RFID reader, it supports players selecting prizes to use tickets on and as such notifying staff that a prize needs to be retrieved while also updating the information in the backend.

The fifth component is the general kiosk which serves all player needs for managing their RFID card and checking data. Namely, this includes issuing a new card, adding tokens to a card, checking token and ticket balances, and viewing selected gameplay statistics associated with the user. These kiosks are present with the mindset of eliminating staff as a human bottleneck within the system, such that players can use general kiosks at their convenience to complete these tasks.

As these two components represent the two specific types of kiosks, they comprise the kiosk subsystem.

3.1.3 Backend Subsystem

The sixth component is the server (or backend) that is the central data hub for the system. It manages the database used to store player, game, and prize information based on the defined data schemas and provides a set of supported requests for machines to use in order to create, read, update, and delete information as needed. All games within the arcade need to be manually registered and programmed with the backend to ensure the data is processed accurately.

The seventh component is the display which serves as a frontend for the server. It is most catered toward the needs of the arcade staff to display useful public-facing information. For example, it can be configured to display top-scoring players for the day or list any out-of-stock prizes in the catalog. This also inherently provides a platform for clients to run promotions based on the displayed data.

3.2 Structure and Relationships

Appendix C contains the Class Diagram intended to clearly illustrate the above definitions and connections among components. For each, attributes, operations, and multiplicities are also given to provide a visualization of the preceding descriptions.

4. Detailed Description of Components

4.1 Component Template Description

The following format shall be used to describe in detail each of the components.

Component Name: **Name of Component**

- *Subsystem:* Subsystem that the component is part of, as shown in the System Architecture diagram of Appendix F.
- *Purpose:* The specific requirements within the Software Requirements Specification [1] which this component satisfies. These are listed in the format FR X.X.X, and as outlined in the Software Requirements Specification [1].
- *Function:* How the component functions to achieve the purpose outlined in the previous field.
- *Subordinates / Modules used:* An outline of the various parts which make up the component.
- *Dependencies:* Which other components this component depends upon.
- *Resources:* The hardware and software resources this component needs to complete the purpose outlined in the Purpose field.
- *Processing:* Full descriptions of the supported functionality from a technical perspective, spanning applicable algorithms and logic.
- *Data:* Data the component is concerned with, as described in section 2.

4.2 Description of RFID Card

Component Name: **RFID Card**

- *Subsystem:* Playing Subsystem
- *Purpose:* FR 2.3.1: Dispense RFID Cards, FR 2.3.2: Add Tokens to Card, FR 2.3.3: Kiosk Display Info, FR 2.3.4: Arcade Game Payment
- *Function:*
    - FR 2.3.1: Dispense RFID Cards: The RFID card is able to store a certain amount of data in various fields. This data can be read by an RFID reader or overwritten by an RFID writer. The data on the arcade RFID cards will be split into the player ID field and the token balance field. In order to dispense a new blank RFID card from the kiosk, the RFID card must support the functionality to have its contents overwritten, setting the player ID field to a unique player ID and the token balance back to 0.
    - FR 2.3.2: Add Tokens to Card: Tokens are then added to the RFID card by incrementing the amount in the token balance field by the amount of tokens being added to the card at the kiosk.
    - FR 2.3.3: Kiosk Display Info: Once the data in the player ID and token balance fields is written, it is stored on the RFID card. Because of this, the data may be read from the card at a General Kiosk in order to display the token balance stored on the card, and the information tied to the player ID stored on the card.
    - FR 2.3.4: Arcade Game Payment: Tokens can also be removed from the RFID card by decrementing the amount in the token balance field by the amount of tokens being spent on playing an arcade game.
- *Subordinates / Modules used:* The RFID card component is made up of a single off-the-shelf RFID readable and writable RFID card.
- *Dependencies:* RFID Reader/Writer

- *Resources:* None
- *Processing:* When participating in read/write operations, the card must return/store the commanded bits on the card.
- *Data:* Stored data include the player ID, which is returned from the backend as a result of a player creation request and is persistent until deleted, and the token balance, an integer with an initial value of 0 that can be incremented by kiosks and decremented by games.

4.3 Description of RFID Reader/Writer

Component Name: **RFID Reader/Writer**

- *Subsystem:* Playing Subsystem
- *Purpose:* FR 2.3.1: Dispense RFID Cards, FR 2.3.2: Add Tokens to Card, FR 2.3.3: Kiosk Display Info, FR 2.3.4: Arcade Game Payment, FR 2.3.5: Win Tickets, FR 2.3.6: Rewards Center Cost
- *Function:*
    - FR 2.3.1: Dispense RFID Cards: The RFID writer is used to overwrite any data which may already be present on an RFID card, providing the user with an RFID card with a uniquely populated player ID field and a token balance field equal to 0.
    - FR 2.3.2: Add Tokens to Card: Whenever a user purchases additional tokens at a General Kiosk, the token balance field on the RFID card is overwritten by the RFID writer to contain the previous balance + the newly purchased tokens.
    - FR 2.3.3: Kiosk Display Info: The user may also have their card read at a General Kiosk in order to check the token balance on the card. In this case, the RFID reader will read the player ID and token balance fields on the card once the card is placed within 50 mm of the reader [2].
    - FR 2.3.4: Arcade Game Payment: In a similar fashion, once the user places their card within 50 mm of an RFID reader on an Arcade Game, the token balance field on the card will be read to determine if the card has enough tokens to play the game. If it does, the token balance field on the card will be decremented by the cost to play the game, by the RFID writer, and the game will begin.
    - FR 2.3.5: Win Tickets: At the beginning of the interaction, the user will have tapped their card at the Arcade Game, allowing the RFID reader on the game to read the player ID field on the card. The game can then use this player ID to update the number of tickets that player has won.
    - FR 2.3.6: Rewards Center Cost: Once a user taps their card at the Rewards Center, the RFID reader on the Rewards Center will read the player ID field on the card, and use this to query the backend for the number of tickets the player has won.
- *Subordinates / Modules used:* The RFID Reader/Writer component is made up of the hardware reader/writer itself and the API for interfacing with the reader/writer. The reader/writer is an off-the-shelf component with a USB interface. The API will be custom

built to allow easy integration of the reader/writer to the General Kiosk, Arcade Game, and Rewards Center software.

- *Dependencies:* RFID Card, Kiosk/Game (to control it)
- *Resources:* Software libraries will be used to ease the development of the API, such as a library for interfacing with the USB device. As the system shall run under Linux, libNFC provides a set of functionalities for interacting with reader/writers like this model. There shall also be an enabled system service that supports these interactions at boot time.
- *Processing:* Read/write interactions with the card happen asynchronously, using interrupt-style event-driven behavior. When a card is brought within range of the reader, it comes out of its idle state and notifies the higher-level application that an card is detected. The program will then either request to read or write data on the card, which results in the respective API call to carry out the operation. Enumerated return values defined in a header file indicate return status of the operation. After completing an operation, the driver places the reader back into its idle state to await further events.
- *Data:* No data are stored on the reader/writer itself, but the API supports the scheme detailed for the RFID Card.

4.4 Description of Arcade Game

Component Name: **Arcade Game**

- *Subsystem:* Playing Subsystem
- *Purpose:* FR 2.3.4: Arcade Game Payment, FR 2.3.5: Win Tickets
- *Function:*
    - FR 2.3.4: Arcade Game Payment: Once the user places their RFID card within 50 mm of the RFID reader, the reader will recognize and read the contents of the card, and the RFID reader/writer API will send the player ID and the token balance contained on the card to the Arcade Game software. The Arcade Game software will first check if the token balance on the card is above the token cost to play the game. If it is, then the player ID will be stored for updating the backend later. Then the game will begin, and then user will play the game. If the token balance is insufficient, a message saying such will appear on the Arcade Game.
    - FR 2.3.5: Win Tickets: Once the user has finished playing the game, they will have received a number of tickets based on their performance in the game. This ticket amount will be displayed to the user. Then, using the stored player ID, this ticket amount, the player ID, and the game ID for this specific game will be sent to the backend for storing and updating the user's profile which corresponds to the player ID.
- *Subordinates / Modules used:* While the design of the game itself will vary depending on the specific type of game, this system will consist of a USB-connected RFID Reader/Writer, network card for handling the connection to the local network, and some user interface for guiding the user through starting a game and displaying statistics after.
- *Dependencies:* RFID Reader/Writer, Server (Backend)

- *Resources:* The hardware requirements of the game itself depend on the nature of its purpose. This should be separated from this system's implementation, however. The Game in the context of this system requires a USB port to connect the RFID Reader/Writer, a NIC to connect to the network (Ethernet preferred for stability), and a physical mounting location for the RFID Reader/Writer. At a minimum, the software requires a single-core CPU with 500MB of RAM to provide comfortable resource allocation space to carry out the above functionality.
- *Processing:* When a card is brought near the reader and it thus wakes from its idle state, the game UI welcomes the player. Using the API for the reader, the token balance is read from the card and compared to ensure that the player can afford the round. Simultaneously, the game requests player statistics from the backend with the read player ID from the card to display any relevant "welcome back" messages. After the player confirms and finishes gameplay, the game publishes statistics to the backend using the temporarily-stored player ID. After the request is served by the backend, the player ID is deleted from the game.
- *Data:* Concerned data for the game is described in section 2.1.

4.5 Description of Rewards Center

Component Name: **Rewards Center**

- *Subsystem:* Kiosk Subsystem
- *Purpose:* FR 2.3.6: Rewards Center Cost, FR 2.3.7: Rewards Center Display Prizes, FR 2.3.8: Rewards Center Alert Staff
- *Function:*
    - FR 2.3.6: Rewards Center Cost:
    - FR 2.3.7: Rewards Center Display Prizes:
    - FR 2.3.8: Rewards Center Alert Staff:
- *Subordinates / Modules used:* All common kiosk components are included (user interface, USB-connected RFID Reader/Writer, and network card for interconnection within the system). The accessory features offered are all software-related and consist namely of the prize catalog.
- *Dependencies:* RFID Reader/Writer, Server (Backend)
- *Resources:* All kiosks require a USB connection to the RFID Reader/Writer and a NIC for connecting to the network (Ethernet preferred). The user interface is preferably touchscreen for easier browsing of the prize catalog. At a minimum, the software requires a single-core CPU with 1GB of RAM to provide comfortable resource allocation space to carry out the above functionality.
- *Processing:* Similar to games, interaction with the system is initiated when a player brings a card in readable proximity of the reader. This triggers an event to be raised within the system, where the application software will call the reader's API function to read the player ID. Once retrieved, a backend request will be constructed for the player's profile as well as the available prize catalog. If applicable, images for the available prizes will be converted from their packed binary format in the backend to be displayed to the player. The player will see their ticket balance at the top of the screen.

When a prize is selected, a request is made to the backend to redeem a prize, which decrements the stock of the prize by 1 and also the ticket balance of the player.
- *Data:* Concerned data for the Rewards Center is described in section 2.3, in addition to the retrieved player information from the backend as described in section 2.2.

4.6 Description of General Kiosk

Component Name: **General Kiosk**

- *Subsystem:* Kiosk Subsystem
- *Purpose:* FR 2.3.1: Dispense RFID Cards, FR 2.3.2: Add Tokens to Card, FR 2.3.3: Kiosk Display Info
- *Function:*
    - FR 2.3.1: Dispense RFID Cards
    - FR 2.3.2: Add Tokens to Card
    - FR 2.3.3: Kiosk Display Info
- *Subordinates / Modules used:* All common kiosk components are included (user interface, USB-connected RFID Reader/Writer, and network card for interconnection within the system). The general kiosk also features a payment terminal for accepting credit card and optionally cash payments to exchange for arcade tokens. This payment terminal is accompanied by a corresponding driver and API that allows eases transaction processing.
- *Dependencies:* RFID Reader/Writer, Server (Backend)
- *Resources:* Same as Rewards Center.
- *Processing:* Similar to games, interaction with the system is initiated when a player brings a card in readable proximity of the reader. This triggers an event to be raised within the system, where the application software will call the reader's API function to read the player ID. Once retrieved, a backend request will be constructed for the player's profile. The menu of the user interface will then present the user with operations. Each will guide the user through the flow of events detailed by the Sequence Diagram found in Appendix D, generally following the pattern of the Rewards Center. The General Kiosk makes similar backend requests to update player information based on the requested operation.
- *Data:* Concerned data for the General Kiosk is described in section 2.2.

4.7 Description of Server (Backend)

Component Name: **Server (Backend)**

- *Subsystem:* Backend Subsystem
- *Purpose:* FR 2.3.5: Win Tickets, FR 2.3.6: Rewards Center Cost, FR 2.3.7: Rewards Center Display Prizes, FR 2.3.9: View Staff Metrics
- *Function:*
    - FR 2.3.5: Win Tickets
    - FR 2.3.6: Rewards Center Cost
    - FR 2.3.7: Rewards Center Display Prizes

    - FR 2.3.9: View Staff Metrics
- *Subordinates / Modules used:* The server software is built on the Node.js framework, which must be installed on the target system. Furthermore, the server should be higher power as compared to kiosks to support scalable traffic loads in peak times of demand within the arcade. For example, a dual-core CPU with at least 2GB would be desirable. If the database is be stored locally on the machine, MongoDB will also need to be installed. The technical documentation for the server designates how to deploy it on a machine.
- *Dependencies:* Arcade Game, Rewards Center, General Kiosk
- *Resources:* The server is the simplest from a hardware perspective. It needs nothing more than a basic runtime environment (see above, Linux preferred) with decent system specs. These are largely governed by the size of the arcade and the number of connected games and kiosks within the system, which would be planned at time of installing.
- *Processing:* The backend supports a variety of tasks and is designed to respond asynchronously to them. When idle, it consumes very few resources until another machine issues a request or a periodic status message is pushed out. All requests take the form of standardized HTTP requests as detailed in the guide for using the backend. JSON is the language used across all requests to transfer data over the wire, as it is portable across platforms and the data are self-describing. Error messages that may result from a request will also comply to the JSON specification in order to provide simple context to the requester.
- *Data:* All supported data schemas that the backend manages are described in section 2.

4.8 Description of Display

Component Name: **Display**

- *Subsystem:* Backend Subsystem
- *Purpose:* FR 2.3.8: Rewards Center Alert Staff, FR 2.3.9: View Staff Metrics
- *Function:*
    - FR 2.3.8: Rewards Center Alert Staff:
    - FR 2.3.9: View Staff Metrics:
- *Subordinates / Modules used:*
- *Dependencies:* Server (Backend)
- *Resources:* The display consists of a small network-connected computer and a connected display. The size of the display should be large enough to allow easy visibility within the arcade. The computer does not require intensive hardware, as it simply accepts messages from the backend and can occasionally make requests. Thus, a single-core CPU with 500MB is more than adequate for this component.
- *Processing:* The display is primarily driven by the backend. In most cases, it should not need to request data. Rather, each request that the backend responds to should also push new player, game, and prize data to the front end, such that the frontend has an up-to-date representation of each data model. When a new message arrives to the

frontend, it will update the displayed content according to a configuration file set up at time of installation, though this can be modified at any time.

- *Data:* Concerned data varies depending on the configuration, but in the extreme case can include everything stored by the backend (though this may become unwieldy in a practical sense).

5. Interface Design

In total, there are 3 subsystems: Playing, Kiosk, and Backend. The user primarily interacts with the Playing subsystem since this consists of the RFID Card, RFID Reader, and Arcade Game components. However, in order to put tokens on the RFID Card, a player must interact with the Kiosk subsystem since this subsystem holds the General Kiosk component. Additionally, the player will eventually interact with the Rewards Center which is also part of the Kiosk subsystem to redeem prizes. Both the Kiosk and Playing subsystems will interact with Backend subsystem since the Playing subsystem pushes and pulls information to and from the Backend, while the Kiosk subsystem performs similar operations.

The Backend subsystem is the centralized interface for communication within the system. All components that request or publish data must comply to the commonized message specification, which in this case is the standard HTTP protocol with message bodies being JSON-formatted. While the player does not see this communication, it is important to emphasize with respect to the interaction among components as well as subsystems.

6. Software Design Goals

The software design goals are the non-functional requirements from the SRS which are as follows: the scanning of an RFID card needs to take less than 3 seconds, the arcade needs to be run by a minimum of 1 staff member, the Backend needs to be able to be updated in less than 3 seconds, and there needs to always be at least one prize available for redemption.

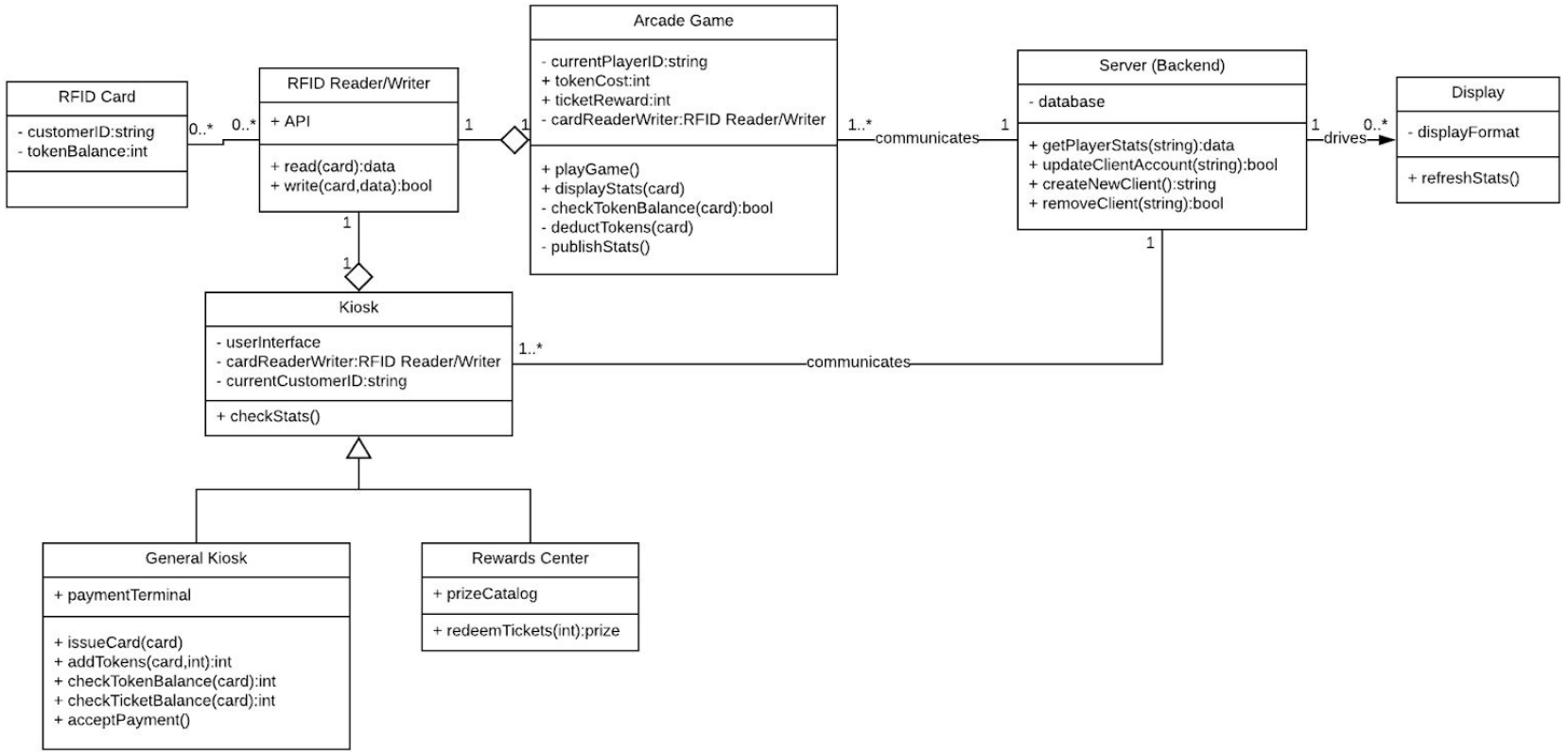
7. Appendices

Appendix A: Scenario

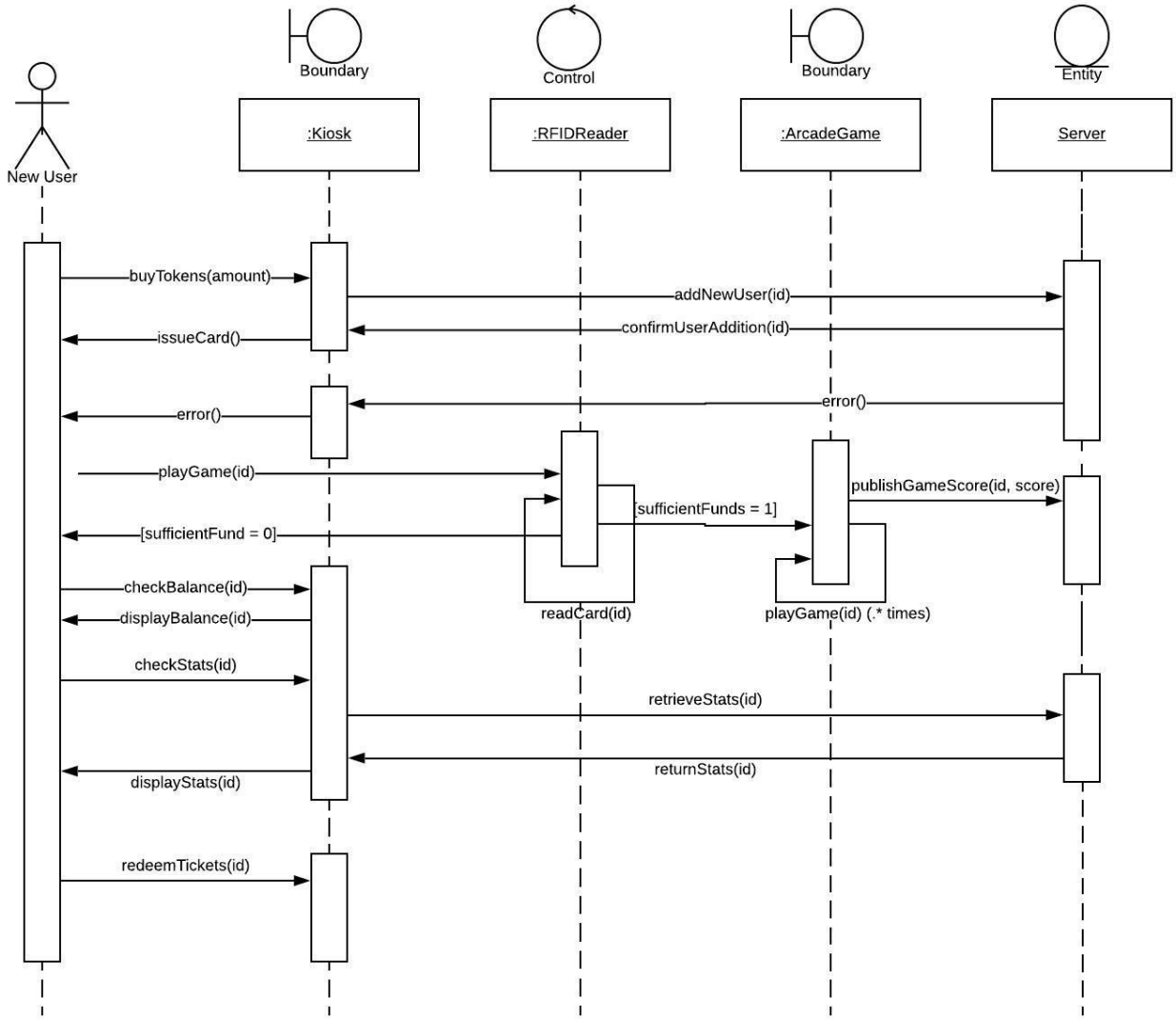
Scenario

- A 12 year-old girl named Miley enters *Global Arcade* for the first time.
- She walks up to a Kiosk and registers herself for a player card, adding $20 worth of tokens to her card.
- The first game that Miley plays is *skee-ball*, where she scans her card on the skee-ball card-reader. Here, she scores the highest score of the day!
- Next she plays *Deal or No Deal* and ended up with case number 14, containing 75 tickets!
- Miley forgets how many tokens she has left and returns to a kiosk where she scans her card and is shown her ticket and token balances. She has 0 tokens left.
- Miley ends her day at the arcade by redeeming her tickets for prizes.
- She also checks to see that she still has the highest *skee-ball* score of the day.

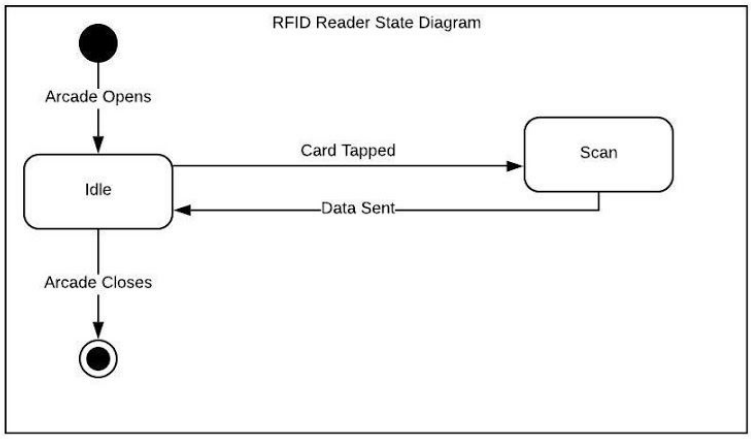
Appendix B: Class Diagram

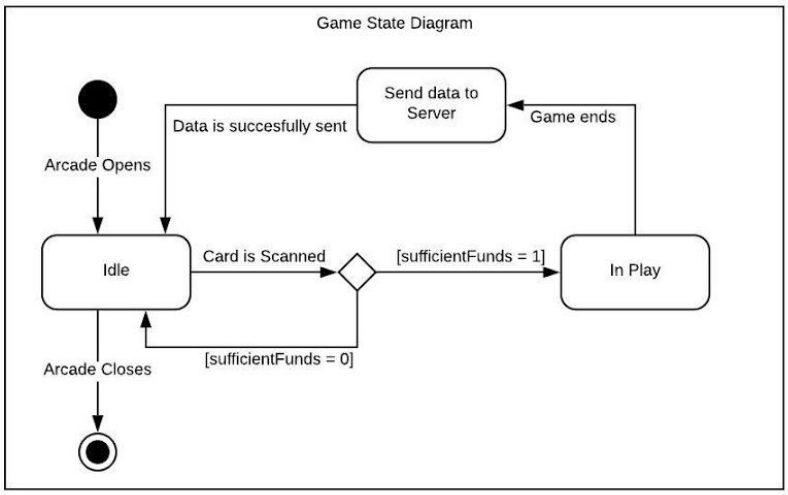


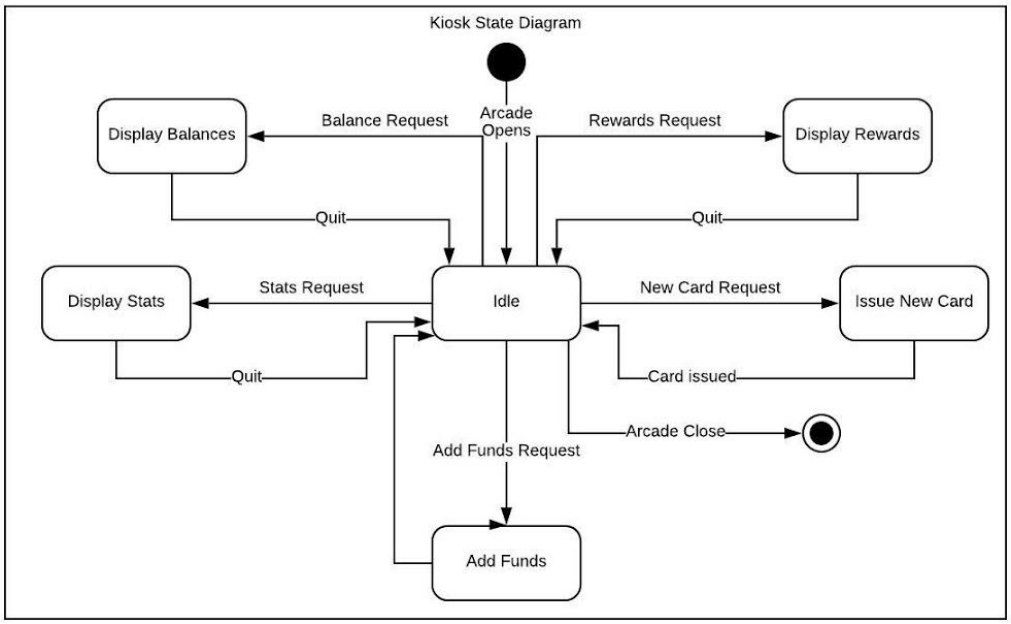
Appendix C: Sequence Diagram



Appendix D: State Diagrams







Appendix E: System Architecture Diagram