CS 130, Homework 2

Will Fehrnstrom, Elizabeth Han, Michael Wang, Michael Wu

UID: 504969404, 004815046, 804862406, 404751542

April 26th, 2019

Part I

Cache Covert Channel Abuse Detection

1 Introduction

1.1 Purpose

Online gaming is a rapidly growing industry with certain games having the potential to reach tens of millions of monthly active users. With its ever-increasing appeal with a mainstream audience, it comes as no surprise than an increasing number of users see merit in abusing the system to gain an undeserved upper hand against others. In some cases these unfair advantages can be translated into monetary gain. At the very least, cheating causes resentment and dissatisfaction with the game leading to reduced user retention. As such, it is of great importance to gaming companies that cheaters be identified and prevented from undermining the experience for others.

This document details the characteristics proposed for implementing a ReplayConfusion Architecture as described by a study performed at the University of Illinois[2]. This system will be used to record suspicious activity on the cloud gaming server which will then be replayed and analyzed to detect

the use of a cache covert channels. This will ideally allow system administrators to collect the evidence necessary to identify this class of cheaters and take the appropriate actions ensuring the integrity of the gaming application for all users.

1.2 Scope

This document is the System Requirements Specification for the proposed covert channel detection system. The requirements cover the functional software components and nonfunctional general constraints the developer will encounter when implementing the new functionality on the gaming server application.

1.3 Intended Audience and Use

This software requirements specification is intended for the client, developers, and end users who want a preliminary overview of the proposed architecture for detecting cache covert channel abuse. It should be noted that this document is the result of a cursory fifteen minute interview with the client and as such is not a complete description of the system or the expected requirements. Developers should not reference this document directly when implementing specific requirements. Rather, it serves as a very high level overview of what the client initially envisioned the system to look like.

1.4 Acronyms and Abbreviations

As this document is intended for a general audience, acronyms and abbreviations will be kept to a minimum.

2 Overall Description

2.1 Product Background and Perspectives

There are a wide range of methods for cheaters to gain an advantage, each usually requiring its own specific countermeasures to identify and address effectively. In this case it is believed that users are cheating via cache covert channels. While users are supposedly limited to communicating with the

server via channels that we control so that only information the user should have access too is transmitted, covert channels allow an alternate route for users to gain access to additional information the server does not intend them to have. As the name implies, cache covert channels transmit this information through the cache. This usually occurs when a trojan process infects the target server. A second malicious process on the outside known as the receiver then primes the cache by filling it with data. The trojan selectively evicts the receiver's data based on the message the trojan wishes to send. Finally, the receiver reads the cache and computes the sender's message based on the number of cache misses.

Unfortunately "There is currently no general detector of cache-based covert channel attacks" [2]. However, the University of Illinois study suggests a novel architecture with the potential to accurately detect a wide range of cache covert channel attacks with "simple and non-intrusive" hardware requirements. Existing defences against these attacks often involve isolating processes or introducing noise to the system muddling any attempted transmissions. These methods alone are insufficient, failing to address all possible attacks or adding significant overhead to the system. The study presents ReplayConfusion as a potential solution. This method employs Record and deterministic Replay to detect covert channel attacks. It takes advantage of the tendency for cache attacks to be highly reliant on "the specific mapping of addresses to cache locations in the machine" and for cache conflicts resulting from these attacks to follow distinctive patterns. By recording the non-deterministic events and cache resulting from a cache covert channel attack and then deterministically replaying the recording using a "different mapping of addresses to cache locations", a new timeline of cache misses can be computed and compared with the original to identify any signs of an attack.

Methods for detecting suspicious behavior and punishing users found to have cheated are already in place. This extension will simply serve as an additional tool for application administrators to collect evidence and build a case against the accused, ensuring the fair treatment of all users.

2.2 User Classes

System Administrators	Administrators shall be able to record and replay activity performed on the server to determine if suspected users are guilty of committing cache covert channel attacks against the cloud gaming server.
Users	Normal users shall be able to use the application without knowing the existence of the ReplayConfusion architecture. Innocent users should never be falsely accused of cheating so false positives are unacceptable. Users will be able to report others as having suspicious behavior. Doing so will cause their match to be recorded and the accused to be flagged. Flagged individuals may then have their future games recorded.
Cheaters	Users attempting to perform cache covert channel attacks shall be detected and identified. As cheaters are likely to be repeat offenders it not paramount that every case of cheating be positively identified. However, the system should be accurate enough that users who are suspected of having performed a cache covert channel attacks on multiple occasions should be confirmed with very high regularity.

2.3 Assumptions and Dependencies

The ReplayConfusion architecture shall interface seamlessly with the currently employed HostelHosting® servers Parts of the RnR (Record and deterministic Replay) Module and Performance Monitoring Unit are already built within the existing system and shall be extended rather than replaced in the new architecture.

2.4 Constraints

1. **Hardware** The system shall be able to run concurrently with gaming applications on cheap multitenanted cloud servers. As such, it shall

introduce little overhead to the server and shall not affect the latency of the gaming applications.

2. **Availability** It is not paramount that every cheater be caught.The system should be able to operate with few interruptions but this constraint is secondary to minimizing performance overheads.

3. **Security** As recordings of games are freely viewable by normal users, it is not the greatest concern that the system secure the secrecy of the data it stores.

2.5 High Level System Decomposition

The diagram below representing the system was copied directly from the the University of Illinois study. In our case the Recorded Apps will be the company's gaming applications. "The shaded boxes show the components of the ReplayConfusion Architecture" [2]. The Cache Address Computation Unit matches the cache with physical addresses. The Cache Configuration Manager allows a system administrator to choose an alternative mapping for the cache address computation unit such that during normal operations the original mapping is used while during replay an alternative mapping unique to the process is used. The Cache Profile Manager collects data of the cache such as number of instructions, accesses, and misses, producing a cache timeline which may be stored and analyzed. The RnR (Record and deterministic Replay) module records non-deterministic inputs to be replayed in a new cache address mapping.

3 System Features & Requirements

3.1 Functional Requirements

3.1.1 General

1. Functional Requirement 1 (ID: GFR1)
   (a) Title: Recording Module
   (b) Required For: Basic Use and Functionality

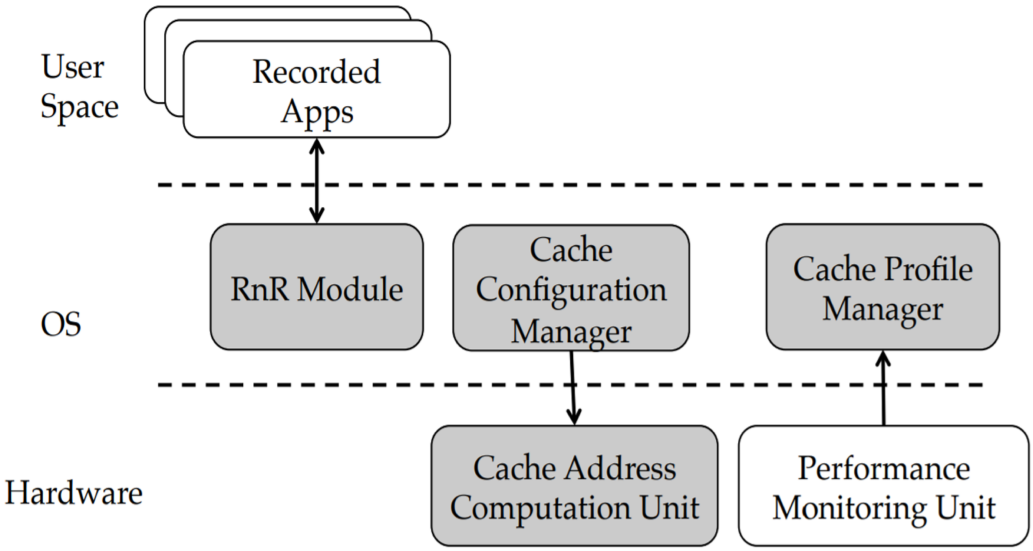

Figure 1: High-level ReplayConfusion architecture, copied from source 2

(c) Description: The existing recording module shall be adapted such that all non-deterministic inputs to the system necessary to perform ReplayConfusion are recorded and stored at a later time. The module shall record the activity of all users during a game which will be discarded after a short time following the game's conclusion unless a user is reported for suspicious behavior, either by another user or by already existing cheating-detection software. The recordings of individuals flagged as suspicious shall be held for a longer period of time, potentially indefinitely.

(d) Dependencies: The HostelHost® server recording module is functioning normally

2. Functional Requirement 2 (ID: GFR2)

(a) Title: Cache Configuration Module

(b) Required For: Basic Use and Functionality

(c) Description: A Cache Configuration Manager OS module shall be produced which employs the HostelHost® server Cache Address Computation Unit allowing application administrators to provide

a process-unique mapping between cache and hardware addresses on demand. The mapping shall have a sufficiently large domain space such that it is highly unlikely that a given cache covert channel attack recording replayed on two randomly produced cache mappings will result in similar cache profiles

(d) Dependencies: The HostelHost® server Cache Address Computation is functioning normally

3. Functional Requirement 3 (ID: GFR3)

   (a) Title: Profile Manager

   (b) Required For: Basic Use and Functionality

   (c) Description: A Cache Profile Manager OS module shall be produced which employs the HostelHost® server Performance Monitoring Unit to record all relevant activity necessary to perform ReplayConfusion and store it for later

   (d) Dependencies: The HostelHost® server Performance Monitoring Uni is functioning normally

4. Functional Requirement 4 (ID: GFR4)

   (a) Title: Cache Analyzer

   (b) Required For: Basic Use and Functionality

   (c) Description: A program shall be produced which takes as input two cache timelines as produced by ReplayConfusion and outputs how likely it is that they are the result of a cache covert channel attack.

   (d) Dependencies: ReplayConfusion has been implemented correctly.

3.1.2 System Administrators

1. Functional Requirement 1 (ID: AFR1)

   (a) Title: Administrator Access

   (b) Required For: Basic Use and Functionality

   (c) Description: System Administrators shall have full access to the features described in this document

   (d) Dependencies: None

3.2 External Interface Requirements

3.2.1 User Interface

The interface of the normal user shall be unaffected by the addition of this functionality. Interfaces for system administrators have no requirements so long as all features described in this document are viewable.

3.2.2 Software Interface

The proposed architecture shall interface seamlessly with the already in place gaming software.

3.2.3 Hardware Interface

The proposed architecture shall interface with the already in place server architecture, with the recording module being extended and Cache Profile Manager operating on top of the existing Performance Monitoring Unit.

3.2.4 Communications Interface

No specification was made regarding the development language, essential libraries, database, or programming environment to be used with the intent being to allow the developers to determine the most cost-effective implementation.

3.3 System Features

3.3.1 Cache Time-line Generation

1. Description & Priority
   Priority 1: System Administrators shall be able to record the non-deterministic inputs to the system resulting from a specific user and produce simultaneously produce a time-line of the corresponding cache activity

2. Stimulus & Response Sequences

   - User has been flagged as suspicious for any reason.

- Record and Deterministic Replay module begins recording non-deterministic inputs to the gaming application originating from the flagged user
- The Cache profile manager produces a history of the cache
- A cache timeline is produced.

3.3.2 Cache Configured Replay

1. Description & Priority
   Priority 2: System Administrators shall be able to replay recordings on a reconfigured cache to produce an alternate cache time-line

2. Stimulus & Response Sequences

   - The system administrator identifies the process recorded and the cache configuration manager automatically computes an alternate cache address mapping.
   - The recording is replayed on the new address mapping
   - An alternate cache timeline is produced

3.3.3 Covert Channel Attack Determination

1. Description & Priority
   Priority 2: System Administrators shall be able to compare two timelines produced from the same non-deterministic inputs under different cache configuration to receive the probability of the user having performed a cache covert channel attack.

2. Stimulus & Response Sequences

   - The system produces two such timelines and feeds it to the timeline comparison program
   - The program outputs the probability

3.4 Non-Functional Requirements

1. Non-functional Requirement 1 (ID: NR1)

   (a) Title: Latency Requirements

(b) Required For: User Experience

(c) Description: The Record and deterministic Replay module system shall not introduce a noticeable increase in latency for users of the gaming application. Other modules are expected to be used during times of low activity and can be allowed to consume more resources.

(d) Dependencies: A minimum viable RnR module has been implemented and is ready for testing.

2. Non-functional Requirement 2(ID: NR2)

(a) Title: Error Rate Requirement

(b) Required For: Basic Use and Functionality

(c) Description: False positives rate should be limited to 0.001% of reported positives. False negatives shall not exceed 80% of actual positive readings

(d) Dependencies: A minimum viable ReplayConfusion system has been implemented and is ready for testing

4 Change Management

The client and development team shall meet once every two weeks during which the client will have the opportunity to express any desire to make changes to the requirements. After a review of the potential effects on the system and the likely cost of implementing the changes, the client will have the opportunity to retract the proposal or proceed at which point the appropriate adjustments to the requirements specification will be implemented. The client has expressed a (potentially unreasonably) high degree of confidence in these requirements and has suggested that any changes presented in the future will only come as the result of an extreme emergency. Such changes should therefore be regarded as paramount and take precedence over the requirements expressed in this document.

5 Document Approvals

Signatory Name	Signatory Position	Date
Michael Wang	Client	4/25/19
Elizabeth Han	Software Developer	4/25/19
William Fehrnstrom	Software Developer	4/25/19
Michael Wu	Software Developer	4/25/19

6 Supporting Information

1. Description of Cache Covert Channels from: `https://cmaurice.fr/pdf/ndss17_maurice.pdf`
2. Description of ReplayConfusion System from: `https://iacoma.cs.uiuc.edu/iacoma-papers/micro16_1.pdf`

Part II
Social Media Parental Controls

1 Introduction

1.1 Purpose

This system is an extension to the functionality of the social media site FaceTime. It implements various optional parental controls providing parents increased awareness and control of their children's account activity.

This document is merely a brief overview of the functionality requested by the client and was compiled over the course of a brief fifteen-minute interview. Developers may refer to this document for a high-level summary of the system but should not rely on it when implementing specific requirements. Concerned parents will find an approachable description of the functions initially envisioned for the system.

1.2 Scope

This document System Requirements Specification for the requested parental controls system. The requirements cover the functional software components and nonfunctional general constraints the developer will encounter when implementing the new functionality on the FaceTime social media platform.

The parent must first have a FaceTime account to link to the child's account. From there the parent can choose among a variety of safety features such as content filtering, account curfews, and automatic alerts in response to friend requests, invitations to chat, etc. Management of the child's safety features is performed via the parent's account.

1.3 Intended Audience and Use

Concerned Parents and their children shall benefit from the increased oversight afforded to parents of their children's FaceTime accounts. System Administrators shall monitor the new system and implement any changes and improvements when necessary.

1.4 Acronyms and Abbreviations

1. FaceTime is the world's leading social media platform. It is the owner and developer of the system described in this document.

2 Overall Description

2.1 Product Background and Perspectives

With the saturation of social media within everyday life, it has become a near necessity for most individuals to have an account to interact with those around them. Whether it's to chat with friends or catch up on the news, users often find themselves spending a significant portion of their day on these sites. However, many parents are concerned that younger users may be at risk when interacting with others over the service. They fear that online predators and the like may attempt to take advantage of children, tricking them into revealing private information or luring them into unsafe situations. In response, FaceTime presents a comprehensive suite of parental controls placing the power and responsibility of safe use of FaceTime by children entirely in the hands of parents.

2.2 User Classes

System Administrators	These users will collect metadata on the new system to help improve the user experience.
Young Users	Users with FaceTime accounts under 18 who are at risk of unsafe online social interactions. They will be placed under the protection of a parent user.
Parents	Users with children with FaceTime accounts. They will be given access to parental controls which help monitor and protect the child's activity on FaceTime.

2.3 Assumptions and Dependencies

It is assumed that both parent and child have FaceTime accounts. This system is built upon an already existing and very well established FaceTime

framework. Any requirements involving changes to the underlying framework are naturally significantly more expensive than requirements that can be built on top of it. As a system built within FaceTime, only activity on a FaceTime account can be monitored. We have no jurisdiction over your child's activity on other sites such a Snapchat or MySpace. Strangers your child encounters in real life are likewise unaffected.

2.4 Constraints

1. The system must be available on all devices and Operating systems which can run FaceTime
2. Tee system must not adversely affect the availability or reliability of the FaceTime platform overheads.
3. The system must not be abusable such that an any user can exercise inappropriately gained parental controls on another arbitrary user.
4. The new features must be available to all users simultaneously upon release.

3 System Features & Requirements

3.1 Functional Requirements

3.1.1 System Administrators

1. Functional Requirement 1 (ID: AFR1)
   (a) Title: Administrator Access
   (b) Required For: User Feedback
   (c) Description: System administrators shall have access to aggregate data about about regarding use of the system. This data shall be aggregated from a large enough dataset as to not compromise the privacy of any individual. For example, administrators may wish to query the average curfew times put in place for children under 16.
   (d) Dependencies: None

2. Functional Requirement 2 (ID: AFR2)

   (a) Title: Administrative Termination

   (b) Required For: Third Party Intervention

   (c) System administrators shall be reserved the right to terminate support for any parental controls for any reason at any time. This includes removing parental controls for specific individuals or for entire principalities.

   (d) Dependencies: None

3.1.2 Parents

1. Functional Requirement 3 (ID: PFR1)

   (a) Title: Parent Request Initiation

   (b) Required For: Basic Use and Functionality

   (c) Description: Users over 18 referred to as the parent shall have the option to send a "Parent request" to another user under 18 referred to as the child. The child can accept this request much like a friend request. Doing so grants the parent access to the parental controls described in this document.

   (d) Dependencies: Both parent and child have and are operating on their own FaceTime accounts

2. Functional Requirement 4 (ID: PFR2)

   (a) Title: Parent Termination

   (b) Required For: Basic Use and Functionality

   (c) Description: The parent shall be able to terminate the parent relationship at any time.

   (d) Dependencies: The parent and child have a current parent-child relationship

3. Functional Requirement 5 (ID: PFR3)

   (a) Title: Curfew Controls

   (b) Required For: Control Enforcement

(c) Description: The parent shall be able to terminate the parent relationship at any time.

(d) Dependencies: The parent and child have a current parent-child relationship

4. Functional Requirement 6 (ID: PFR4)

   (a) Title: Parent newsfeed access

   (b) Required For: Monitoring

   (c) Description: The parent shall have access to the child's newsfeed and the home pages of all users the child has access to.

   (d) Dependencies: The parent and child have a current parent-child relationship

5. Functional Requirement 7 (ID: PFR5)

   (a) Title: Parent message access

   (b) Required For: Monitoring

   (c) Description: The parent shall have access to the child's newsfeed and the home pages of all users the child has access to.

   (d) Dependencies: The parent and child have a current parent-child relationship

6. Functional Requirement 8 (ID: PFR6)

   (a) Title: Parent message access

   (b) Required For: Monitoring

   (c) Description: The parent shall be able to view all friend requests and invitations to message being sent to and from the child. Additionally, the parent will be alerted each time the child receives a message. The contents of this message shall not be viewable.

   (d) Dependencies: The parent and child have a current parent-child relationship

7. Functional Requirement 9 (ID: PFR7)

   (a) Title: Content Filtering

(b) Required For: Control Enforcement

(c) Description: The parent shall be able to place a filter on the child's account which will automatically remove certain words or phrases.

(d) Dependencies: The parent and child have a current parent-child relationship

3.1.3 Children

1. Functional Requirement 10 (ID: CFR1)

   (a) Title: Accepting Parent Request

   (b) Required For: Basic Use and Functionality

   (c) Description: he child can accept this request much like a friend request. Doing so grants the parent access to the parental controls described in this document. As the parental relationship is vastly different from the friend relationship, multiple warning screens will be presented to anyone attempting to accept a parent request.

   (d) Dependencies: Both parent and child have and are operating on their own FaceTime accounts

2. Functional Requirement 11 (ID: CFR2)

   (a) Title: Child Terminating Relationship

   (b) Required For: Basic Use and Functionality

   (c) Description:Once the child turns 18 they shall from that point onward have the option to terminate the parent relationship.

   (d) Dependencies: None

3.2 External Interface Requirements

3.2.1 User Interface

The child's interface shall be unaffected by the presence of a parent relationship accept when parental controls such as curfews are being enforced. The parent's interface shall include intuitive controls that are inline with FaceTime's design philosophy and should reuse user interface designs when possible. For example, notifications indicating the child has received a friend request shall appear similar to a normal friend request.

3.2.2 Software Interface

The software must be integrated seamlessly into the FaceTime platform. No previous functionality shall be impeded by the addition of these features.

3.2.3 Hardware Interface

The additional functionality must be enclosed in and interface with the same servers and databases already being used by FaceTime.

3.2.4 Communications Interface

Specifications regarding the development language, libraries, database, or programming environment are left flexible to encourage developers to find the best possible implementation. However, developers should default to using the tools employed in the development of similar features in the rest of the FaceTime feature suite.

3.3 System Features

3.3.1 Child Connection and Control

1. Description & Priority
   Priority 1: The Parent shall be able to set up a parent-child relationship with a child and initiate parental controls.

2. Stimulus & Response Sequences

   - The parent logs into their FaceTime account and sends a parent request to a child user.
   - The child logs into their FaceTime account and accepts the parent request. The child user will be presented with multiple warnings explaining clearly the risks involved in accepting the request.
   - The parent on their FaceTime account gains access to a new set of settings allowing for monitoring and parental control of the child's account.
   - The parent receives notifications allowing for passive monitoring of any attempts to communicate with the child over FaceTime.

3.4 Non-Functional Requirements

1. Non-functional Requirement 1 (ID: NR1)
   (a) Title: Reliability Requirements
   (b) Required For: User Experience
   (c) Description: The reliability of the FaceTime platform shall be unaffected by the addition of these features. As this system is merely a simple extension of already existing features, it is expected that failure in the parental controls would be equivalent to failure of the entire FaceTime system.
   (d) Dependencies: A minimum system has been implemented and is ready for testing.
2. Non-functional Requirement 2 (ID: NR2)
   (a) Title: Security Requirement
   (b) Required For: Security
   (c) Description: The security of user's data shall be unaffected by the addition of these features with the acknowledgement that the parent may unintentionally leak the private information of their child.
   (d) Dependencies: A minimum viable system has been implemented and is ready for testing
3. Non-functional Requirement 3 (ID: NR3)
   (a) Title: Privacy Requirement
   (b) Required For: Privacy
   (c) Description: The system shall not reveal any information about the child's account that has not been explicitly stated by this document. In particular, the parent shall be have login permissions to the child's account.
   (d) Dependencies: A minimum viable system has been implemented and is ready for testing
4. Non-functional Requirement 4 (ID: NR4)

(a) Title: Legality Requirement

(b) Required For: Legal Compliance

(c) Description: The legality of certain features may be contested in some principalities. Certain controls shall be able to be disabled in such principalities without affecting the functionality of other controls

(d) Dependencies: A minimum viable system has been implemented and is ready for testing

5. Non-functional Requirement 5 (ID: NR5)

   (a) Title: Control Enforcement Latency

   (b) Required For: User Experience

   (c) Description: controls such as curfews and content filters shall be enforced before any content is made viewable on the child's device.

   (d) Dependencies: A minimum viable system has been implemented and is ready for testing

6. Non-functional Requirement 6 (ID: NR6)

   (a) Title: Notifications Latency

   (b) Required For: User Experience

   (c) Description: notifications to the parent shall occur no later than 1 second after the child receives the notification.

   (d) Dependencies: A minimum viable system has been implemented and is ready for testing

4 Change Management

The client shall receive semi-weekly updates of the progress of the system, during which the client will have the opportunity to express any desire to make changes to the requirements. After a review of the potential effects on the system and the likely cost of implementing the changes, the client will have the opportunity to retract the proposal or proceed at which point the appropriate adjustments to the requirements specification will be implemented. Cost is not a major concern to the client so such reviews shall focus

mainly on guaranteeing the feasibility of the changes rather than computing the minutiae of resource requirements.

5 Document Approvals

Signatory Name	Signatory Position	Date
Michael Wang	Client	4/25/19
Elizabeth Han	Software Developer	4/25/19
William Fehrnstrom	Software Developer	4/25/19
Michael Wu	Software Developer	4/25/19

Part III
Secure Classified File Sharing

1 Introduction

1.1 Purpose

The purpose of this file sharing service is the following: service FBI agent and officials requests' in a highly authenticated manner; using many-factor authentication in order to share confidential case files between FBI offices. There is a distinct need to share files in a more efficient manner than by shipping paper copies, which are vulnerable to theft, loss, and tampering. The FBI's internal communication channels are already in place and are secure: therefore, we need to plug into those communication channels in order to send our files. Servers will be distributed across different field offices to prevent any single point of attack. Furthermore, the files will primarily still be kept on paper, but will be shared by scanning, and then deleted after sending.

1.2 Scope

This project will focus on the sharing of currently on-paper reports, with an emphasis on the access control required to do so. Specifically out of scope are any other messages on the platform other than the distribution of official documents. Further, we are targeting field offices, so this project is not expected to conform to any other office requirements. The physical infrastructure that we will put our software on is explicitly in scope at the clients' request. In fact, we are responsible for procuring that hardware. In addition, the methods of agent authentication in order to scan documents and transfer them are in scope-we will find a suitable, PC compatible iris scanner.

1.3 Intended Audience and Use

The FBI office staff, officers, and agents. This software specification should also be viewable by any penetration testers. That is, allowing them to view this is beneficial in exposing system vulnerabilities.

1.4 Acronyms and Abbreviations

1. FBI: Federal Bureau of Investigation, Government Agency for prosecuting US internal affairs.
2. Agent: a highly authorized human entity tasked with carrying out the FBI's mission.
3. IT: Internet Technology
4. AES: Advanced Encryption Suite
5. Client: The FBI agency.
6. Development Team: The contracted development team, working on this product.

2 Overall Description

2.1 Product Background and Perspectives

Previously, FBI offices in the field were prevented from effectively coordinating with each other on one case because of the difficulties in distributing the case information (on paper) necessary. The intent of this software product is to solve this problem by digitizing papers for a short period of time for distribution. As the problem domain involves foreign intelligence agencies and officers charged with penetrating the US's internal monitoring system, the FBI, we should expect that any flaw in the software that we place will ultimately be exploited by the adversary. Therefore, security is of the highest importance. We must ensure that only officers assigned to a specific case are allowed to send and receive documents relating to that case. Further, documents should be distributed to all offices involved in a case at their request, and not be distributed to offices that have no part in a case. We must plan for negligence, i.e. officers leaving our software open in order to deal with some more pressing matter. No matter the organization, office workers will tend to let their guard down.

2.2 User Needs

FBI Agent	This user class comprises all of the FBI agents that will be using our product in the field to help distribute case information between offices.
Government officials	Periodically, government officials outside the FBI department may wish to see this software system in use as part of an audit. Appropriate security procedures should be followed in this situation to provide minimal access.
Foreign Operatives	This user class is a user class in a very non-traditional sense: they will likely examine our product closely to find any exploits: planning is required to make exploiting the system as difficult as possible for this user class.
FBI IT Management	The FBI's internal IT management team will need to maintain the system after we've built it, thus they will invariably become the primary consumers of our documentation created during the development process, and they will have to have the most intricate understanding of the hardware, software, and communications interconnection. We will be providing to them all our research and development documents, and further, we would like to allow auditing of the system response to user load, without giving away any confidential information.

2.3 Assumptions and Dependencies

An assumption that cannot be made about our operating environment is that it will be secure. We must assume that any bad actor can at any time gain physical access to our system. We assume that any agent authenticated by our authentication scheme is a good actor: it is the job of the authentication equipment to ensure that this is the case. Furthermore, we assume that the installation environment of the system will be a good one-that is, no bad actors will be present for the installation process. This is a key assumption, but one that must be taken if we are to roll out the system at all. We assume that the client's requests are in good faith, and that they too are

good actors. Furthermore, since our own developers are subject to intense scrutiny, we assume that they too are good. We depend heavily on the hardware platform that we use. We also depend on the electrical systems currently place in FBI field offices, and the internal wired network that the FBI uses in order to communicate and fetch information for the wider web. We also depend on the transportation infrastructure used to put the hardware in place.

2.4 Constraints

1. A minimum number of external intrusions must be allowed.
2. Document sharing must be achievable by one entity with a network connection, provided they have appropriate authorization.
3. The system must be portable to a wide variety of locations with varying network access.
4. The system must be recoverable in the event of electrical failure.
5. Multi-factor authentication must be used.
6. The system must only be used within FBI field offices.
7. Authentication classes should be hardwired, and unconfigurable.
8. The hardware required per office must take up no more than 10 cubic feet.
9. The hardware required per office must be able to be shipped in a variety of different formats (it must be durable).

3 System Features & Requirements

3.1 Functional Requirements

Unless otherwise stated, every functional requirement have the following dependencies:

1. Transport Network

2. Electrical Network

3. FBI Communications infrastructure

Priorities are ranked from level 1 (PR1) (imperative) to level 3 (PR3) (Add-On Feature)

3.1.1 User Class A - FBI Agent

1. Functional Requirement 1 (ID: AFR1) (PR2)

   (a) Title: Accessibility

   (b) Required For: Basic Use and Functionality

   (c) Description: The software system MUST be accessible directly from any field agent's desk. The authentication systems used to permit use of the system must be similarly accessible, or be multiplied to an extent that allows direct use. The system must not be accessible to unintended users.

2. Functional Requirement 2 (ID: AFR2) (PR1)

   (a) Title: Authentication

   (b) Required For: Basic Use and Functionality

   (c) Description: The software system MUST not be usable without first completing all forms of authentication enumerated: iris scanning, physical security token, and passcode.

3. Functional Requirement 3 (ID: AFR3) (PR1)

   (a) Title: Document Scanning

   (b) Required For: Field Office Communication

   (c) Description: The software system MUST be able to scan a variety of paper documents into a digital format, and only store the resulting digitized form for a maximum of one minute.

4. Functional Requirement 4 (ID: AFR4) (PR1)

   (a) Title: Document Sending

   (b) Required For: Field Office Communication

(c) Description: The software system MUST be able to send digitized documents to the desired field offices along a pseudo-randomized path. Document sending should be knowingly initiated and document receipt confirmed.

5. Functional Requirement 5 (ID: AFR5) (PR2)

   (a) Title: Automatic Sign-Out

   (b) Required For: Security

   (c) Description: The software system MUST automatically sign out a field agent after a maximum of 40 seconds of idle activity. The sign out must not be recoverable, and must only be circumvented by a full sign in procedure.

6. Functional Requirement 6 (ID: AFR6) (PR3)

   (a) Title: Obfuscation

   (b) Required For: Security

   (c) Description: The software system's internal workings must be obfuscated from dissection from a disassembler or related software programs that attempt to reverse engineer potential exploits from underlying machine code.

7. Functional Requirement 7 (ID: AFR7) (PR2)

   (a) Title: Adding New Users

   (b) Required For: Scalability

   (c) Description: It should be a trivial process for a user with sufficient permissions to add another permitted user to the pool of authorized users. This new user MUST be constrained to have an authorization level lower than the user granting permissions.

3.1.2 User Class B - Government Officials

1. Functional Requirement 1 (ID: BFR1) (PR2)

   (a) Title: Server Audit

   (b) Required For: FBI Accountability Mandates

(c) Description: Because security is of the utmost importance, we cannot allow API hooks within the application for fear that they might be misused. Therefore, the server should be able to generate a traffic report of the number of sensitive requests through the server since the last audit, as well as other various useful statistics, such as the number of times that users were automatically logged out for idling, and any service abnormalities that occurred.

2. Functional Requirement 2 (ID: BFR2) (PR1)

   (a) Title: Audit Authentication

   (b) Required For: FBI Accountability Mandates

   (c) Description: Government officials should be subject to the same strict level of security we require for FBI agents to use the software. However, we do not possess retinal data for government officials. Therefore, government officials SHOULD be provided with a temporary physical security token on their arrival at FBI office premises, and they SHOULD be made to enter a secure password autogenerated at a cryptographic level of randomness.

3.1.3 User Class C - Foreign Operatives

1. Functional Requirement 1 (ID: CFR1) (PR1)

   (a) Title: Proactive monitoring

   (b) Required For: Security

   (c) Description: On any tampering with the physical server or upon mass document scanning from any given workstation, the system MUST send out an internal FBI communications message to the FBI Director, FBI IT manager, and the director of the specific field office being tampered with should also be notified.

3.1.4 User Class D - FBI IT Management

1. Functional Requirement 1 (ID: DFR1) (PR3)

   (a) Title: Documentation Portal

   (b) Required For: Maintainability

(c) Description: We MUST provide an offline site for internal use by the FBI IT team, stocked with detailed documentation about the system from an architectural view, design view, and also construction view.

2. Functional Requirement 2 (ID: DFR2) (PR2)

   (a) Title: Auditing System

   (b) Required For: FBI Internal Self Audits

   (c) Description: FBI IT team members should be able to access the same information as that provided to external department government auditors, and furthermore, the information should be portable from the server using flash drive.

3. Functional Requirement 3 (ID: DFR3) (PR1)

   (a) Title: Internal IT Authentication

   (b) Required For: FBI Internal Self Audits

   (c) Description: FBI IT team members MUST provide the same level of multi-factor authentication as FBI agents in order to grab audit data from the server.

4. Functional Requirement 4 (ID: DFR4) (PR3)

   (a) Title: Internal IT Audit Rates

   (b) Required For: FBI Internal Self Audits

   (c) Description: FBI IT team members MUST NOT be subject to any rate limiting in terms of the audits that they are allowed to perform. However, audits themselves should contain information about how many audits have been performed: an audit of audits, so to speak.

3.2 External Interface Requirements

3.2.1 User Interface

The user interface should present a convenient way to scan documents after authentication, and send already scanned documents. The sign out button

should be prominently shown, and the user should be warned 10 seconds before they are due to be signed out because of a lack of activity. The sensitive file contents being scanned should be viewable from a separate window of the application, but should not be viewable by default from the main entry point. The user interface should warn users if they download documents from the application to their computer, in order to prevent leakage of sensitive information.

3.2.2 Software Interface

The software described herein should contain no external API hooks. It must be built from the ground up using an FBI vetted programming language and any external library use must be on the FBI whitelist of approved frameworks and libraries. The software interface should be modularized to allow maximum extensibility, except in regards to authentication code, which should be left unchanged to the maximum extent in order to promote security. On change of the software interface's authentication methods, FBI IT management should be notified, and the FBI director responsible for overseeing this project should also be contacted to give approval.

3.2.3 Hardware Interface

The hardware server must be enclosed within a secure housing inaccessible physically to personnel except for IT specialists. It must be permanently connected to a power source that is hardened with a backup generator.

3.2.4 Communications Interface

The software produced must conform to the internal FBI communications specification, which will not be enumerated here for security reasons, but can be viewed in a separate document. The communications from client to server must be encrypted before being sent through the interface, and encryption should be AES level or higher.

3.3 System Features

3.3.1 Document Transmission & Receipt

1. Description & Priority
   Priority 1: The user must be able to send scanned documents over the FBI network to other FBI offices.

2. Stimulus & Response Sequences

   - Clicking file scan.
   - Clicking file send after file scan complete.
   - Having document receipt confirmed.

3.3.2 Audit Generation

1. Description & Priority
   Priority 2: Certain User Classes must be able to reliably obtain an audit of the system's usage and any abnormal behavior observed.

2. Stimulus & Response Sequences

   - Pressing a physical button on the server.
   - Plugging into a flash-drive port on the server.

3.4 Non-Functional Requirements

1. Non-Functional Requirement 1

   (a) Title: Communication Scarcity

   (b) Required For: Security

   (c) Description: The software system MUST NOT send any extraneous information besides that contained within the document and that which is essential for packet routing to external FBI offices.

2. Non-Functional Requirement 2

   (a) Title: Credentials

   (b) Required For: Security

(c) Description: The software system SHOULD NOT reveal the credentials of the logged on user in any way to any entity other than the controlling authentication agent present in the server.

3. Non-Functional Requirement 3

   (a) Title: Reliability

   (b) Required For: Robust Software

   (c) Description: The software system SHOULD remain up for at least 99.999999% of the time. Failure of the central server in a given FBI field office should not affect other FBI field offices.

3.5 Design Constraints

1. Account information and case information should not be cached. Computer memory locations holding sensitive information should be flushed after use.

2. Document communication must take less than one minute end-to-end, from sending to receiving at a field office.

3. This software should be develop in a safe and robust programming language, Rust. Rust is also approved by the FBI for use in developing internal applications.

4. Hardware should not have to be reconfigured on power failures, on maintenance downtime, and on office lockdown.

5. Authentication classes should be set at the beginning of the product's deployment, and not configurable short of a new software version altogether.

4 Change Management

The FBI will provide vetted requirements specialists always available and internal to the agency that will work with the contracted software company in order to ensure that the best deliverables are pursued. On any requirements change, the signature of the FBI IT manager is required. There will be a

formal meeting every two weeks to track the progress of the software product. In order for the meeting to take place, the FBI IT manager must be present to give comment, along with an internal auditor, the requirements specialists, and a representative three software developers of the contracted company. After software release, the change management procedure should go through the IT manager and the representative of the FBI Operations Department.

5 Document Approvals

Signatory Name	Signatory Position	Date
William Fehrnstrom	IT Manager	4/25/19
Elizabeth Han	Software Developer	4/25/19
Michael Wang	Software Developer	4/25/19
Michael Wu	Software Developer	4/25/19

Part IV
Glucose Monitor Wireless Data Transfer

1 Introduction

1.1 Purpose

The purpose of this software is the following: transfer glucose monitoring data to a remote database. Diabetic patients are normally given monitors to track their blood sugar level. However, all the collected data is stored locally, which makes it difficult for medical personnel to access when they need in order for an accurate diagnosis. The files would have to be manually sent over by the client, which could induce a time delay in the analysis the doctor would produce after looking over the data. A tightly controlled server will need to be introduced to automatically send the data in real time.

1.2 Scope

This product will focus on the sharing of locally stored data on continuous glucose monitoring systems, as well as the authentication to make sure that patients have control over who is accessing their data. This product will not focus on analysis of the actual glucose data beyond the very basic, i.e. glucose levels being dangerously high.

1.3 Intended Audience and Use

The diabetic patients and the physicians/care providers. Patients should be able to view their real-time and stored data – physicians and caregivers may view this data after patients give permissions.

1.4 Acronyms and Abbreviations

1. Patient: a diabetic using glucose monitoring software in order to maintain their health

2. Physicians/Care providers: the medical personnel in charge of monitoring the patient's health and providing their professional opinion and diagnoses
3. Development Team: The contracted development team working on this product
4. HTTP: Hypertext Transfer Protocol.
5. TCP: Transmission Control Protocol.
6. POST Request: A request made to a server to update its data.
7. GET Request: A request made to a server to retrieve information from it.

2 Overall Description

2.1 Product Background and Perspectives

Continuous glucose monitoring systems already provide a way for patients to see their glucose levels in real time, as well as view trends. However, they do not possess the capability to let physicians view this data. Patient privacy is of great importance: only medical personnel assigned to the specific patient should be able to view this data, and only if the patient gives continuous permission for them to see it. The one exception is if the patient exceeds acceptable glucose levels; the aforementioned medical personnel will NEED to see this information to ensure the patient's health.

2.2 User Needs

Patient	This user class comprises all of the FBI agents that will be using our product in the field to help distribute case information between offices.
Physician	This user class consists of all the doctors who will be viewing glucose data and receiving it on a consistent basis.

2.3 Assumptions and Dependencies

We assume that the patient is in possession of a glucose monitoring device. We assume that any user that logs in is, in fact, that user: it is patient and physician responsibility to keep their usernames/passwords private. We also assume that the physician always has internet connection, courtesy of their place of employment. We assume that the client will have internet connection at least once every 24 hours.

2.4 Constraints

1. The patient must give their consent to allow the physician to access their personal glucose data, otherwise the patient's glucose data must not be revealed to any third party without their knowledge.
2. The system must take some action given a patient's high glucose level.
3. The glucose monitoring device has only 10 Megabytes of storage.
4. The patient may sometimes enter areas without a network connection.

3 System Features & Requirements

3.1 Functional Requirements

3.1.1 User Class A - General

1. Functional Requirement 1 (ID: AFR1) (PR1)
   (a) Title: Transfer
   (b) Required For: Data Transfer to Physician
   (c) Description: If an internet connection is available, all measurements taken by the glucose monitor shall be automatically transferred wirelessly to a centralized database. This information is linked with the patient's account and is usually only viewable by the patient unless the patient explicitly shares this information or under extreme circumstances.
2. Functional Requirement 2 (ID: AFR2) (PR1)

(a) Title: Storage

(b) Required For: Completeness of Data

(c) Description: If an internet connection is unavailable, the glucose monitor shall store the most recent readings locally, discarding the oldest readings if the memory limit is exceeded. Upon reconnecting to the internet, the glucose monitor shall prioritize sending the oldest readings on record.

3. Function Requirement 3 (ID: AFR3) (PR1)

(a) Title: Analysis

(b) Required For: Preliminary Warning

(c) Description: The database shall be capable of performing a preliminary analysis of the readings such as detecting life threatening readings or trends, but this preliminary analysis should not go beyond basic detection and notification.

3.1.2 User Class B - Patient

1. Functional Requirement 1 (ID: BFR1) (PR1)

(a) Title: Permissions

(b) Required for: Legal Concerns

(c) Description: From their accounts, patients shall be able to give access to their readings based on date of measurement to their care provider and physician of choice. Patients can choose additional physicians and care providers at any time.

2. Functional Requirement 2 (ID: BFR2) (PR1)

(a) Title: Personal Account

(b) Required for: Ease of Use

(c) Description: Patients shall be able to log on to to a personal account to view all past readings. UI elements such as graphs shall be provided to aid in the visualization of the data.

3. Functional Requirement 3 (ID: BFR3) (PR2)

(a) Title: Inoperability Notice

(b) Required for: General use

(c) Description: Patients shall know when their devices are malfunctioning via device notifications. If malfunctions inhibit the device notifications, the device should perform a soft shut down.

3.1.3 User Class C - Physicians/Care providers

1. Functional Requirement 1 (ID: CFR1) (PR1)

   (a) Title: Permissions

   (b) Required for: Legal Concerns

   (c) Description: Physicians and care providers shall not have access to patient measurements unless given explicit permission by the patient or unless the readings meet certain criteria (defined by the monitor manufacturer) indicating the patient's health to be in danger. Readings that meet these criteria shall be automatically shared with the patient's physician and care provider even without explicit permission.

2. Functional Requirement 2 (ID: CFR2) (PR1)

   (a) Title: Patient Alert

   (b) Required for: Real-Time Wellness

   (c) Description: Physicians and care providers should be notified when a patient's glucose level spikes to dangerous levels. They should be able to be notified through the following communication channels: email and text.

3. Functional Requirement 3 (ID: CFR3) (PR3)

   (a) Title: Patient History

   (b) Required for: Data Review

   (c) Description: Physicians and care providers should be able to view a graph of patient glucose levels over time in real-time.

3.2 External Interface Requirements

3.2.1 User Interface

A UI should be present on the display of the patient's glucose monitor and should inform the user of high glucose levels, as well indicate when the system has become inoperable in an alarming font and icon. There should be a UI for viewing glucose data on a web portal. On the Web Portal, patients will be given a prompt to sign in to be able to use the system. Once they have signed in, they should immediately see any alarm notifications as cards underneath a graph displaying their glucose levels. The physician should have access to the same UI, once granted permission by the patient.

3.2.2 Software Interface

The software interface provided by the glucose monitoring device should contain operations for retrieving data in a given span of time, and retrieving data points above a certain threshold. The web portal should operate on an analysis API run from a remote server that processes user glucose levels and performs common analytics on them.

3.2.3 Hardware Interface

The hardware interface given by the glucose monitoring device should allow for networked communication, temporary storage of glucose data, and should be operable under operating conditions from -10°F to 130°F.

3.2.4 Communications Interface

The Communication Interface for the glucose monitor shall be TCP. HTTP POST/GET requests will be made to the analytics server from client computers (physicians and patients)

3.3 System Features

3.3.1 Data Transmission

1. Description & Priority
   Priority 1: Users must be able to access real-time records of the glucose levels of the patient

2. Stimulus & Response Sequences

    - Log into account
    - (if physician) Select patient name
    - Click view records
    - Have records displayed, possibly in graphical format

3.3.2 Privacy

1. Description & Priority
   Priority 2: Patients must be able to reliably control who is able to access their data

2. Stimulus & Response Sequences

    - Automatic notification after a certain period of time, asking patient to give permission for physician to view data
    - Click 'Yes' or 'No'
    - Have permissions confirmed or denied

3.4 Non-Functional Requirements

1. The service must have medical rated reliability: uptime must be more than 99.9999999%

2. The service must have reasonable response time: under normal operating conditions, it should take a maximum of a minute from a new glucose data point to server processing. Under dangerous conditions, where a user is in a life threatening situation, system total response time must be under 15 seconds.

3. The service must be secure: multi-factor authentication should be used. At the very least, strong password protection must be mandated for every user.

4 Change Management

At the beginning of the development process, focus groups of physicians and glucose sensitive patients should be assembled, and asked to participate in a semi-regular product feedback loop during the development process. These product feedback meetings should take place every month at the end of an agile sprint. Developers should program in pairs.

5 Document Approvals

Signatory Name	Signatory Position	Date
Elizabeth Han	Medicorp Director	4/25/19
Will Fehrnstrom	Software Developer	4/25/19
Michael Wang	Software Developer	4/25/19
Michael Wu	Software Developer	4/25/19

Part V
Windows Upgrade Application for Bank

1 Introduction

1.1 Purpose

The client is a bank that has spent money developing software that works on Windows 7. Our client wishes to create an application that helps upgrade their code to be compatible with Windows 10. This should be done safely, easily, and automatically so that the client has minimal hassle with the upgrade process.

1.2 Scope

Our application will be an IDE that performs automatic code conversion, formatting, and and refactoring. It will also generate diffs in order to help the bank know what has been changed. We will add tools to refactor various parts of the code such as changing variable names and function names.

1.3 Intended Audience and Use

The two main types of people who will be dealing with our software will be the bank's developers and the bank's testers. The developers want to have an easy to use editor that changes code quickly and error free, while the testers want an easy way to see what has been changed and why. The developers will be able to edit in a graphical environment that includes a "convert" function which automatically detects and fixes any code sections that are incompatible with Windows 10. The testers will be able to look at logs generated by the "convert" functionality that indicates what was changed.

1.4 Acronyms and Abbreviations

1. IDE: Integrated Development Environment

2. UI: User Interface
3. Diff: A log that shows the difference between a file before and after changes
4. Client: The bank

2 Overall Description

2.1 Product Background and Perspectives

Our application makes it easy to convert code that works on Windows 7 to Windows 10. It automatically detects any outdated code constructs and replaces them with workarounds that are compatible with Windows 10. This enables developers to quickly get their applications working on the new platform with minimal hassle. The code conversion process should only take a single click for the entire project. Developing this system will save our clients the cost of having humans manually converting every file in a large code base.

2.2 User Needs

Developers	This user class comprises all of the bank's employees that will be using our product to convert their code to use Windows 10 compatible code.
Testers	This user class consists of the bank's employees that will be looking at the logs that our product generates and verifying that the code is compatible with Windows 10.

2.3 Assumptions and Dependencies

An assumption that we make is that the code is easily readable by a machine and able to be converted to Windows 10. Sometimes code upgrades require sophisticated workarounds that only a human can perform. In this case our product will simply show an error message that a conversion could not be performed. Then a human can fix the problem. Additionally we assume that the main issues with the upgrade are code incompatibilities due to different functions, naming conventions, and other constructs that can be parsed. Then our IDE can use algorithms similar to those used in compilers to figure

out what code constructs are being used. We assume that our client's application uses standard object oriented coding languages such as those related to Java and C++, so we do not need to develop a converter for every possible language. We can expand the covered languages at a later date.

2.4 Constraints

1. We are given a set of source files and must produce an updated set of source files that is compatible with Windows 10.
2. The input source files are compatible with Windows 7.
3. Sometimes code changes cannot be converted to be compatible and our system must detect this.
4. The system must work on the languages that are used in our client's applications.

3 System Features & Requirements

3.1 Functional Requirements

Priorities are ranked from level 1 (PR1) (imperative) to level 3 (PR3) (Add-On Feature)

3.1.1 User Class A: Developers

1. Functional Requirement 1 (ID: AFR1) (PR1)
   (a) Title: Convert
   (b) Required For: Automated Changes
   (c) Description: The IDE should look through the source code and swap out any incompatible code with new code that works with Windows 10.
2. Functional Requirement 2 (ID: AFR2) (PR2)
   (a) Title: Refactor
   (b) Required For: Manual Changes

(c) Description: The IDE should be able to perform actions such as renaming variables, changing function names, modifying function definitions, formatting code, and rearranging functions so that developers can easily make manual code changes.

3. Functional Requirement 3 (ID: AFR3) (PR1)

   (a) Title: Error Detection

   (b) Required For: Manual Changes

   (c) Description: The IDE should be able to highlight any incompatible code sections that cannot be converted so that developers can see where they need to make manual changes.

4. Functional Requirement 4 (ID: AFR4) (PR3)

   (a) Title: Source Explorer

   (b) Required For: Ease of Use

   (c) Description: The IDE should be able to show a tree that contains the source files in the project. This allows developers to easily navigate through the code base.

3.1.2 User Class B: Testers

1. Functional Requirement 1 (ID: BFR1) (PR2)

   (a) Title: Diffs

   (b) Required For: Logging

   (c) Description: The IDE should generate a diff that shows changes made after converting code. This way testers can verify that the code works and add comments to a change log.

2. Functional Requirement 2 (ID: BFR2) (PR2)

   (a) Title: Help Messages

   (b) Required For: Logging

   (c) Description: The IDE should be able to explain any changes with a help message.

3.2 External Interface Requirements

3.2.1 User Interface

The UI will be a graphical interface that displays the project structure of the code that a developer is working on. There should also be an editor window that lets the developer make changes to the source code. There will be a menu option to convert the code to be compatible with Windows 10.

3.2.2 Software Interface

Our software will be a standalone application that runs locally on the client's machines. Because of this, there will be no API calls to any external sources or any other software components apart from the IDE executable.

3.2.3 Hardware Interface

Our software will be compiled to run of the machines that our client uses for development. As our application only runs on a single machine at a time, no other hardware support is needed. We may need two versions of our IDE, but only if our client specifies that they develop on two incompatible platforms such as Linux and Windows.

3.2.4 Communications Interface

Our IDE will initially not perform any communication, though we may later add in a feature to perform updates over the network.

3.3 System Features

3.3.1 Code Conversion

1. Description & Priority
   Priority 1: The user must be able to automatically convert code to be compatible with Windows 10.

2. Stimulus & Response Sequences

   - Load project into IDE.
   - Click to convert code to be compatible with Windows 10.
   - Show a results screen with any possible errors or success status.

3.3.2 Log Generation

1. Description & Priority
   Priority 2: After the conversion some sort of log should be generated.

2. Stimulus & Response Sequences

   - The developer clicks to convert code.
   - A log file is saved in the file system with a diff and descriptions of changes.

3.4 Non-Functional Requirements

1. Non-Functional Requirement 1

   (a) Title: Execution Time

   (b) Required For: Speed

   (c) Description: The IDE should be able to update the code in $O(n^2)$ time or better in order to ensure that the code updates take a reasonable amount of time.

2. Non-Functional Requirement 2

   (a) Title: Multiple Language Support

   (b) Required For: Flexibility

   (c) Description: The IDE should be able to convert code in multiple languages so that all the code in the client's applications can be converted.

3. Non-Functional Requirement 3

   (a) Title: Simple Interface

   (b) Required For: Ease of Use

   (c) Description: The IDE should be easy to navigate and learn so that developers can quickly begin to use it.

3.5 Design Constraints

1. We will develop the application as a lightweight standalone application. This way it can be quickly distributed.

2. The application should focus on the minimum functionality in order to get the conversion process started quickly.

3. We will design tokenizers and parsers for each language and create an abstract syntax tree in order to process the code.

4. We will research the code differences in Windows 10 and Windows 7 in order to make the appropriate transformations on the code. These changes will be hard coded into our application.

4 Change Management

Changes will go through the client first. For each change we will have a meeting and draft a change proposal in order to document the reasons why we are making a change. During our change meetings we will also specify the additional costs of making a change.

5 Document Approvals

Signatory Name	Signatory Position	Date
Michael Wu	Client	4/25/19
Elizabeth Han	Software Developer	4/25/19
Michael Wang	Software Developer	4/25/19
William Fehrnstrom	Software Developer	4/25/19