

Credit Karma Finance Systems Software Development Life Cycle (SDLC)

V1.1

July 6, 2017

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Notes** |
| 1.0 | July 6, 2017 | Initial version |
| 1.1 | July 7, 2017 |  |

Table of Contents

[1. Introduction 4](#_Toc487202080)

[2. Definitions 4](#_Toc487202081)

[3. SDLC Process 4](#_Toc487202082)

[a. Resources 4](#_Toc487202083)

[b. Sprint process 5](#_Toc487202084)

[4. SDLC Process Controls 5](#_Toc487202085)

[a. Strategic Direction 5](#_Toc487202086)

[b. Program Development and Program Change 5](#_Toc487202087)

[a. Security 5](#_Toc487202088)

[b. Availability 7](#_Toc487202089)

[c. Code development process 8](#_Toc487202090)

[d. Code deployment 8](#_Toc487202091)

[e. Code quality 9](#_Toc487202092)

[f. Processing integrity 9](#_Toc487202093)

[5. Appendix A: AGILE SCRUM Documentation 10](#_Toc487202094)

# Introduction

This document details the IT general controls in Finance department software development and operation. Finance Systems follows the AGILE SDLC methodology. Appendix A contains links to AGILE documentation.

# Definitions

|  |  |
| --- | --- |
| Term | Definition |
| **AGILE** | A software development methodology emphasizing iterative software development. |
| **Credit Karma (CK)** | Credit Karma Inc. and corporate employees, systems, and processes. |
| **SCRUM** | A daily meeting as part of the AGILE methodology to review team members’ current tasks and blockers. |
| **Software Development Life Cycle (SDLC)** | The process used to develop, test, verify user acceptance of, deploy, operate, and end-of-life software. |
| **Sprint** | A time period of one month or less in which the team develops a specific set of deliverables. |

# SDLC Process

## Resources

The Finance Systems SDLC process consists of the following resources:

|  |  |
| --- | --- |
| Resource | Role |
| **Engineering Management** | Decide on requirements and relay to Development Team lead. |
| **Team Lead** | Lead Development Team to implement requirements. |
| **SCRUM Master** | Manage SCRUM process during Sprints. |
| **Development Team** | Implement requirements. |

## Sprint process

In the AGILE methodology, development is based around sprints. All resources except management are present at all stages of the sprint process. The Sprint Process is as follows:

1. Engineering Management provides requirements to Team Lead.
2. Hold Sprint Planning meeting to communicate requirements to team members and plan the implementation of the requirements.
3. Team begins work on implementation and holds daily SCRUM meetings.
   1. In SCRUM meetings, team members communicate their current tasks and blockers.
4. Changes to Sprint plan are made as needed throughout the sprint, based on new requirements and information.
5. At end of sprint, a retrospective meeting is held to evaluate success and areas of improvement for the previous sprint.
6. A sprint backlog meeting is held to establish which tasks from the previous sprint remain to be completed.
7. A sprint planning meeting is held to start the next sprint.

# SDLC Process Controls

## Strategic Direction

The Finance Systems Steering Committee is responsible for setting organizational goals, OKRs, and strategic objectives for Finance Systems. The Committee approves new projects, ensures that projects align with strategic business requirements, and ensures that projects utilize approved tools and technologies.

## Program Development and Program Change

## Security

When a new requirement is received, team leads are responsible for listing all security risks associated with a new requirement. The team lead is responsible for minimizing any potential vulnerabilities in a software deployment. These risks include, but are not limited to:

* Physical security risks
  + Natural disasters and accidents
  + Datacenter facility security breaches
  + Theft of multifactor authentication devices
  + Theft of physical credentials
  + Unauthorized access by current and past employees
  + Unauthorized access by external competitors
  + Unauthorized access by other attackers
* Application, network, and social engineering security
  + IP spoofing
  + Inadequate firewalling
  + Inadequate data segregation
  + Inadequate role segregation and separation of duties
  + Social engineering attacks
  + Man in the middle attacks
  + DNS poisoning
  + Software exploits
    - Trojans
    - Worms
    - Viruses
    - Malware
    - Rootkits
    - Cracks
    - Ransomware
  + Cross site scripting
  + SQL injection
  + Eavesdropping
    - Network sniffing
    - Intellectual property theft
  + Insecure key and password transfer
    - Password cracking
    - Password breach
    - Flawed password recovery processes
  + Data transfer risks
  + Phishing
  + Tunneling
  + Backdoors
  + Replay attacks
  + System tampering
  + System penetration
  + TCP/IP hijacking
* Combinations of multiple attack vectors
* Protect Credit Karma Inc. intellectual property
  + Secure communication for all source code and associated resources
  + Use of secure, private GitHub repositories
    - Administrative access limited to GitHub Admin

Responsibility for enforcing compliance with stated risks is with Security Team. Each team is responsible for incorporating security safeguards, precautions, and contingency planning in accordance with the following security control policies:

* Credit Karma Inc. (CK) services initiate all transactions with external web services.
  + External web services never initiate transactions with CK systems.
* Only the minimal necessary ports are opened. All other ports on application and web servers are closed off and firewalled.
* All interactions with open internet are done through a Squid proxy web server.
* Passwords must be transmitted through secure encrypted medium and destroyed after receipt.
* Centralized secure credential storage in Pleasant Password Server.
* Fire Eye DNS and mail protection using DKIM authentication.
* Email whitelisting.
  + Whitelisting of partner email addresses and system service accounts
  + Blacklisting of all other emails to Credit Karma addresses.
* Thorough penetration testing, debugging, load testing, and feature testing are performed before deploying systems to production
* Thorough and detailed integration testing, unit testing, and feature testing to ensure protection against all threats and attack vectors mentioned in (4)(a).

## Availability

Credit Karma systems maintain 100% 24/7x365 availability by application of the highest standard of software best practices. There practices include:

* Clustering
* Use of high quality distributed processing and data storage tools
  + Airflow
  + Big Query
  + Spark
* Network load balancing
* Service level agreements with external SaaS services
* Monitoring
  + Splunk
  + Elastic Search
  + Slack alerting
  + Application logging
  + Server logging
* Process to report and resolve security incidents and suspicious activity reports
* Full recovery plan
  + Regularly tested with latest system software deployment configurations
  + Detailed instructions for recovery from catastrophic failure stored in Confluence for each team project
  + Ensure that additional IT hardware is available in case of complete hardware and datacenter failure
  + Hard-copy recovery manuals, IT process manuals, and application bootstrap procedures
* High quality software infrastructure
  + IT Steering Committee approves software choices per security, quality, and availability standards
  + Select well supported active commercial and open-source software tools and frameworks in use by Fortune 500 information technology companies
  + Extensive error detection and coding at all stages of application flow
  + Each server has separate network cards with separated public and private connectivity
    - Keep public and private connectivity separate in application layer in addition to physical layer
  + Synchronize all system and application clocks at all times
  + Ensure data backup using RAID and high quality, professionally certified data backup technologies
  + Require latest version of PGP, RSA, SSL, SFTP, OpenAuth2, SSL, and all other protocols in use
  + Update all system software and operating system tools on a scheduled basis to ensure that all software is updated to latest version using apt-get, yum, and standard operating system tools
* Regularly review all security plans on a monthly basis as well as when any system architecture or application changes are made, or a production deployment as made, to ensure completeness, quality, and clarity.
* All employees are to have up-to-date certification all technologies they are responsible for utilizing, operating, or implementing.

## Code development process

* JIRA ticket for feature or bug fix request is created by product customer
* Team Lead assigns resources to ticket during Sprint Process
* Assigned resources fulfill feature request and commit code to developer GitHub forks
* All GitHub commits must include descriptive comments mentioning ticket number
* Ticket is updated during development as new information becomes available
* Unit tests are created for all code
* Developers pull request completed code to “develop” branch of repository
* Team lead approves all pull requests
* Jenkins automatically builds “develop” branch and deploys to development environment
* Team thoroughly tests code in development environment using integration testing, load testing, feature testing, and penetration testing
* Team lead pushes code for completed features to “QA” branch
* Jenkins automatically builds “QA” branch code and pushes to QA environment
* Development Team demos completed product to customer using QA environment
* Customer confirms acceptance of product and acknowledges completion of feature
* Team lead pushes QA branch code to “Staging” branch
* Jenkins automatically builds “Staging” branch code and pushes to Staging environment
* Development Team performs final testing in staging environment
* Team lead pushes completed code to production
* Customer confirms that code is working in production
* Customer closes ticket
* Staging environment is maintained as failover in case of production failure

## Code testing

All code is to be subject to the following testing:

* Load testing
* Unit testing
* Integration testing
* Penetration testing
* Feature testing
* User acceptance testing

All tests are to be kept up to date with 100% coverage of latest features.

## Code deployment

* Latest master code versions continuously pushed to production in continuous integration flow
  + Latest code for an application is pushed to Master branch of appropriate repository in Credit Karma GitHub instance
  + Jenkins job is triggered via GitHub Hook to build master branch
  + Jenkins pulls latest artifacts from code artifact repositories
    - Jenkins builds code from GitHub
    - Jenkins pushes latest compiled artifacts to artifact repo
    - Jenkins deploys latest compiled code executables to production
    - Jenkins performs deployment tasks
      * Delete deployments of obsolete code and executables
      * Push and run latest executables to production
  + Traceability to be maintained using:
    - JIRA Tickets
    - GitHub commits, merges, and operations
    - Jenkins deployment logs
    - Server logs
    - Application logs

## Code quality

All systems will be architected, designed, and planned by the Team Lead in coordination with Team resources. The Development Team will create system architecture documentation with detailed diagrams, schematics, and data flows for all system features. The Team Lead is responsible for ensuring that the final product matches the planned architecture. The Team Lead is responsible for approving any architecture changes in response to changing requirements, information, system issues, or hardware needs. The system

Finance Systems will ensure code quality using the following practices:

* Maintainability
  + Code is to be fully documented with descriptive commit messages, fully documented system architecture, and system design documentation
  + Class, variable, function, method, and other data and code names are to be descriptive, concise, and accurate
  + High quality software frameworks are to be used to ensure decoupling, traceability, error reporting, and debugging functionalities
  + Documentation is to be updated with any system changes, emphasizing any breaking changes
* Dependability
  + Software product is to be thoroughly tested per the standard defined in (4)(b)(d)
  + Team Lead is to assign on-call resources 24/7x365 for all production systems
  + Thorough testing in development, QA, and Staging environments before production deployment
  + Continued monitoring of production systems after deployment
* Efficiency
  + Software team is to exclude unneeded files, data, code, and feature sets from final product
  + Code is to comprehensively utilize latest multithreading and distributed processing frameworks to ensure full hardware resource utilization
  + Code is to be thoroughly tested for bugs and issues causing latency, throughput loss, and performance issues
* Usability
  + Code is to be verified with end customer to confirm ease of use, completeness, and quality fulfillment end user requirements
  + Debugging and technical data is to be logged separately and never exposed to end user
  + User interface should include to the necessary information to efficiently complete all user requirements and use cases

## Processing integrity

Finance Systems obeys the following practices in regards to processing integrity and data integrity to ensure complete, authorized, valid, and accurate transaction processing:

* Adherence to high standards of code quality defined in 4(b)(f)
* Comprehensive evaluation of all transaction data
* Centralized source of truth tracing:
  + All transactions
    - State
    - Receipt Time
    - Read Time
    - Processing Start Time
    - Processing End Time
    - Processing Completion Status
    - Processing Failure Status
    - Processing Metadata
  + Regularly scheduled jobs are to access the central transaction table
    - For all failed, idling, or missed process and transactions, jobs will:
      * Detect
      * Reschedule
      * Re-iterate
      * Confirm completion
      * Log:
        + Status
        + Number of retries
        + Errors
* Highest quality application framework, authentication, and security protocols standards are to be used as specified in (4)(b)(a) to ensure only authorized users access processes and/or data
  + Clear separation of duties in user roles
  + Limited privileges granted based on user role needs
  + Clear and comprehensive logging and alerting at all stages of authentication
* Transactions are to be validated against expected use cases
  + Comprehensive anomaly detection and alerting
  + Monitoring of ongoing transactions to ensure valid state and latency of all operations and transforms
  + Visualization, reporting, and monitoring software to conform to the high standards set in (4)(b)(a) and (4)(b)(b)

# Appendix A: AGILE SCRUM Documentation

SCRUM Guide documentation. http://scrumguides.org/scrum-guide.html