**RCrypt**

Encrypted Resilient Private Cloud Storage



Produced For:

Dr. Taehyung Wang

CSUN COMP 380 – Introduction to Software Engineering

Spring 2018

18 May 2018

Produced by: Group 5

Trishen Patel

Curtis Noonan

Muhammad Hassan

Michael Thorin

Jesus Salinas

Table of Contents

[Project Description 3](#_Toc514445138)

[Requirements and Specification 4](#_Toc514445139)

[Use Case Diagram and Descriptions 4](#_Toc514445140)

[Use Case Diagram 4](#_Toc514445141)

[Use Case Descriptions 5](#_Toc514445142)

[Functional Requirements 18](#_Toc514445143)

[Non-Functional Requirements 20](#_Toc514445144)

[Design 21](#_Toc514445145)

[System Modeling 21](#_Toc514445146)

[Context Modeling 21](#_Toc514445147)

[Process Modeling 22](#_Toc514445148)

[Interaction model 23](#_Toc514445149)

[Behavioral Modeling 24](#_Toc514445150)

[Structure Modeling 25](#_Toc514445151)

[System Architecture and Patterns 27](#_Toc514445152)

[4+1 views 28](#_Toc514445153)

[User View 28](#_Toc514445154)

[Process View 30](#_Toc514445155)

[Physical View 32](#_Toc514445156)

[Development View 34](#_Toc514445157)

[Logical View 34](#_Toc514445158)

[Detailed Design Principles 35](#_Toc514445159)

[Testing 36](#_Toc514445160)

[Project/Process 36](#_Toc514445161)

[Open Issues 36](#_Toc514445162)

[Retrospective 37](#_Toc514445163)

# Project Description

Data security is an issue not readily addressed in small to medium sized businesses. The aim of this project is to develop a robust file encryption and storage software which could ostensibly be sold to businesses. The unspoken purpose of this project is to develop a software product which is unique and challenging such that this project could be a talking point on a resume.

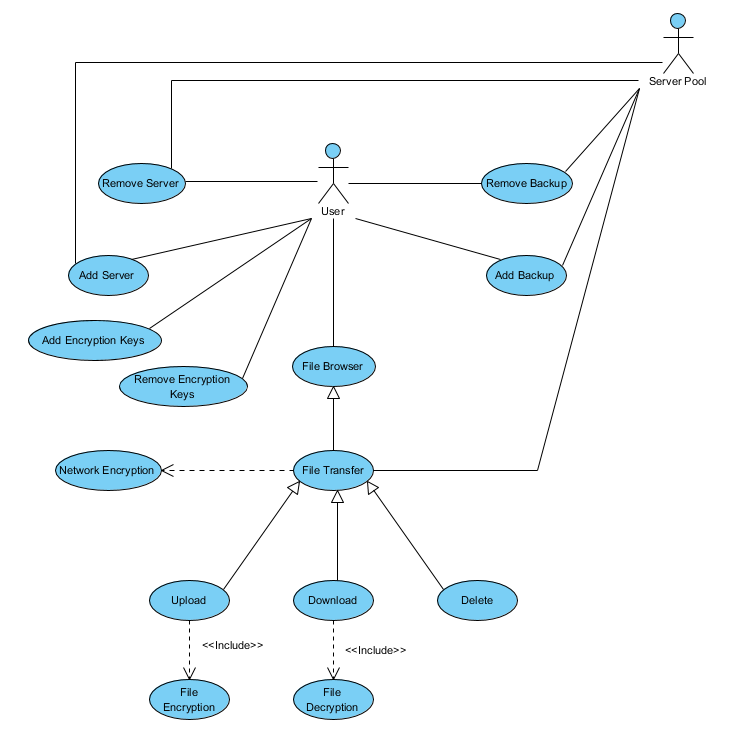
The scope of this project encompasses socket programming, encryption, UX, and scheduling. For socket programming, the software sends both file data and java objects across a wide area network. A travelling event log is one of the objects sent to each server. There are algorithms which group multiple remote servers into logical “replication pools” and distribute the files between the remote servers within the pool. For encryption, the files are encrypted using symmetric AES 128 bit. To maintain file ambiguity, the file names are changed to a unique pool-specific string which is then encoded to a string of hexadecimal bytes. To the average user the file names would look like gibberish, and to an expert user who decodes the hexadecimal, the file names would not reveal any information about the contents of the file. Instead, the list of encoded file names and their real names are stored in an encrypted java object. Each replication pool is assigned a symmetric encryption key which is used to both encrypt files before sending and decrypt files after receiving. These encryption keys are generated with only a random seed to make them more secure, and are also stored in the encrypted java object. The detailed information of each pool, scheduled backups, and event logs are also stored in the same encrypted java object, called the user file. When logging in, the user’s username is hashed and this hash is used to look up the user file name, and the user’s password is salted and then used to generate a symmetric encryption key. This key is the same key every time the same password is entered. The user file is decrypted using this key and loaded into memory. Upon exiting the program or logging out, the user file is encrypted using the user’s key and saved to the computer’s disk. The user interface follows the analogy of multiple panes of glass in a single window. To create a user interface we combined FXML files to specify the layout of our components with controller classes to control each of those FXML files. Using Scene Builder 2.0 we modified the FXML files to reflect our UI design and imported the JFoenix library for side menu functionality. Utilizing JavaFx functionality in collaboration with Scene Builder 2.0 and JFoenix we created a functional user experience that allows navigation through windows with file operations being called from the UI view. For the backup scheduling, we allow the user to choose multiple files, or folders, and back them up to a server pool of their choice. It can be their personal server, or work server, or any other server of their choice. We also allow the user to schedule a backup one time, daily, weekly, or monthly.

The stakeholders of this product at this juncture are only the professor and the students of group 5.

# Requirements and Specification

## Use Case Diagram and Descriptions

### Use Case Diagram



### Use Case Descriptions

Add Backup

Description – User adds a scheduled or continuous back to a selected server pool.

|  |  |
| --- | --- |
| Scenario | A user adds a backup to a server pool. |
| Triggering event | User clicks “add backup” button. |
| Actors | User, Server Pool |
| Related use cases | N/A |
| Stakeholders | User |
| Pre-  condition | A server needs to exist for it to be backed up. |
| Post-  condition | The server has been backed up. |
| Flow of  events | |  |  | | --- | --- | | Actor | System | | 1. Click “add backup” button. | 1. Creates a backup of server. 2. Adds backup to server pool. | |
| Exception | ShapeStep Condition Action Description  2a Backup not Sets an alarm and notifies user that backup was not created.  Created and asks user to try again.  3a Backup not Sets an alarm and notifies user that backup was not added  added to pool to server pool. |

Remove Backup

Description – Removes a existing backup of a server pool.

|  |  |
| --- | --- |
| Scenario | A user will remove a backup. |
| Triggering event | User clicks “remove backup”. |
| Actors | User, Server Pool |
| Related use cases | N/A |
| Stakeholders | User |
| Pre-  condition | Backup needs have been created and backed up in pool. |
| Post-  condition | A backup is removed from server pool. |
| Flow of  events | |  |  | | --- | --- | | Actor | System | | 1. Clicks “remove backup” button. | 1. Finds backup from server pool. 2. Removes backup from server pool. | |
| Exception | ShapeStep Condition Action Description  2a Backup not Sets an alarm and notifies user that backup was not found  Found and asks user to try again.  3a Backup not Sets an alarm and notifies user that backup was not  removed removed. |

Add Server

Description – Adds a server to pool.

|  |  |
| --- | --- |
| Scenario | User adds a server to the pool. |
| Triggering event | User clicks "add server" button. |
| Actors | User, Server Pool |
| Related use cases | N/A |
| Stakeholders | User |
| Pre-  condition | Server needs to exist. |
| Post-  condition | There will be a server that will be available to use. |
| Flow of  events | |  |  | | --- | --- | | Actor | System | | 1. Click add server. 2. Choose server | 1. Adds server to pool. | |
| Exception | ShapeStep Condition Action Description  3a Failure to add Set an alarm and notify user that server was not added.  server |

Remove Server

Description – Removes server a from the server pool.

|  |  |
| --- | --- |
| Scenario | User removes a server from the pool. |
| Triggering event | A user wants to remove a server from the pool. |
| Actors | User, Server Pool |
| Related use cases | N/A |
| Stakeholders | User |
| Pre-  condition | Server needs to be in the pool. |
| Post-  condition | There is one less server in the pool. |
| Flow of  events | |  |  | | --- | --- | | Actor | System | | 1. Click “remove server”. 2. Choose which server to remove | 1. Finds the server needed to be removed. 2. Removes server. | |
| Exception | ShapeStep Condition Action Description  3a Server not Set an alarm and notify that no such server exists and ask  found user to try again.  4a Server not Set an alarm and notify that the server was not removed.  removed |

Add Encryption Key

Description – Creation and storage of an encryption key for encrypting and decrypting files

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario** | A user wants to use different encryption keys for different files, and must first create the different keys | | |
| **Triggering event** | User clicks "add key" button | | |
| **Actors** | User | | |
| **Related use cases** | None | | |
| **Stakeholders** | User | | |
| **Pre-**  **condition** | Key storage location must be online | | |
| **Post-**  **condition** | Additional encryption key available for encrypting files | | |
| **Flow of**  **events** | |  |  | | --- | --- | | **Actor** | **System** | | 1. "add key" button pressed. | 1. System generates a 2048-bit encryption key 2. System saves encryption key in a secure location | | | |
| **Exception** | Step | Condition | Action Description |
| 2a  3a. | System fails to generate a new key  Key storage is not available | Set an alarm and ask user to try again.  Set an alarm and inform user that storage location has been disconnected. |

Remove Encryption Key

Description – User wants to remove a key used for encrypting/decrypting files

|  |  |
| --- | --- |
| **Scenario** | A user removes an encryption key, which prevents it from being used for further encryption/decryption. |
| **Triggering event** | User clicks “remove encryption key” button |
| **Actors** | User, Server Pool |
| **Related use cases** | None |
| **Stakeholders** | User |
| **Pre-**  **condition** | Key Storage must be online. |
| **Post-**  **condition** | Encryption key removed. |
| **Flow of**  **events** | |  |  | | --- | --- | | **Actor** | **System** | | 1. “remove encryption key” button clicked | 1. Encryption key removed from storage | |
| **Exception** | Step Condition Action Description  2a. key storage not online Set an alarm and notify user that key storage is not online |

Upload

Description – A file is uploaded to a server pool when the user is encrypting a file.

|  |  |
| --- | --- |
| **Scenario** | A file is selected by the user to encrypt |
| **Triggering event** | * User selects files and presses the “upload” button * Backup schedule triggers an upload |
| **Actors** | User, Scheduled Backup |
| **Related use cases** | Depends on Encryption, Inherits from File Browser |
| **Stakeholders** | User, Scheduled Backup, Destination Server |
| **Pre-**  **condition** | Files selected in File Browser |
| **Post-**  **condition** | Files will be encrypted and copied on to the selected server |
| **Flow of**  **events** | |  |  | | --- | --- | | **Actor** | **System** | | 1. Clicks the upload button  2. Select server location and location on server | 2. File browser window appears  4. Encrypts file and transfers it to the server | |
| **Exception** | Step Condition Action Description  2a file browser does not appear Restart the application  4a file not found List a “no file found” error  4a location not found List a “location does not exist” error |

Download File

Description – User wants to download a file from the server pool.

|  |  |
| --- | --- |
| **Scenario** | A user wants to download a file from the server pool onto his/her computer. |
| **Triggering event** | 1. User clicks "download a file" button. 2. User selects files to be downloaded. 3. User clicks "download selected files" button. |
| **Actors** | User |
| **Related use cases** | None |
| **Stakeholders** | User |
| **Pre-**  **condition** | Files must exist on the server pool. |
| **Post-**  **condition** | System will not change. |
| **Flow of**  **events** | |  |  | | --- | --- | | **Actor** | **System** | | 1. "Download a file" button is clicked. 2. User selects files to be downloaded. 3. "Download selected files" button is clicked. | 1. System highlights files. 2. System sets a check on files selected. 3. System downloads files which were selected. | |
| **Exception** | Step Condition Action Description  2b. No files found. Notify the user that there are no files are found. |

Delete File

Description – User wants to permanently delete a file.

|  |  |
| --- | --- |
| **Scenario** | A user deletes a file from the server pool. The user will be given the choice to delete one or more files. |
| **Triggering event** | 1. User clicks "delete a file" button. 2. User selects files to be deleted. 3. User clicks "delete selected files" button. |
| **Actors** | User, Server Pool |
| **Related use cases** | None |
| **Stakeholders** | User |
| **Pre-**  **condition** | Files must exist on the server pool. |
| **Post-**  **condition** | Selected files will be erased from server pool. |
| **Flow of**  **events** | |  |  | | --- | --- | | **Actor** | **System** | | 1. "Delete a file" button is clicked. 2. User selects files to be deleted. 3. "Delete selected files" button is clicked. | 1. System highlights files. 2. System sets a check on files selected. 3. System deletes files which were selected. | |
| **Exception** | Step Condition Action Description  4a. No files found. Notify the user that there are no files are found. |

File Transfer

Description – Copies the contents of one file location to another file location

|  |  |
| --- | --- |
| **Scenario** | A file is duplicated to a memory address |
| **Triggering event** | The user clicks on a file in the file browser |
| **Actors** | User  Server Pool |
| **Related use cases** | File Browser, Upload, Download |
| **Stakeholders** | User |
| **Pre-**  **condition** | Secure Socket File Stream must be created between User’s local machine and a Server in the Server pool  Enough storage space must be available on local drive or server |
| **Post-**  **condition** | The file will have been transferred to a remote server from the local machine, or visa versa. |
| **Flow of**  **events** | |  |  | | --- | --- | | **Actor** | **System** | | 1. User chooses a file to upload or download 2. User receives transfer status 3. User receives file transfer confirmation | 2. File is compressed  3. Secure socket is opened between client and server utilizing network encryption  4. Transfer begins  5. Transfer completes and socket is closed | |
| **Exception** | Step Condition Action Description  3a. Network Error Inform user of the connection error  3b. Server Not Responding Inform user to check the server/try again  4a. Not enough storage space Inform user to increase storage capacity  4b. Network connection interrupt Inform user to try again |

File Browser

Description – The file browser appears when a user is encrypting or decrypting a file. It allows the user to select the location of the upload, download, or delete use cases.

|  |  |
| --- | --- |
| **Scenario** | A window appears with a list of the files on the computer and a list of files on the server |
| **Triggering event** | The user encrypts or decrypts a file |
| **Actors** | User |
| **Related use cases** | File Transfer |
| **Stakeholders** | User |
| **Pre-**  **condition** | No file browser must currently be open |
| **Post-**  **condition** | A window is open that lists local files and encrypted server files |
| **Flow of**  **events** | |  |  | | --- | --- | | **Actor** | **System** | | 1. The user selects encrypt or decrypt  3. Select location to copy or delete | 2. Window appears with list of local and remote files  4. Encrypt or decrypt, copy or delete | |
| **Exception** | Step Condition Action Description  2a no file to transfer List a “no file selected” error  4a encryption key not found List a “encryption key error” error |

Encryption

Description – Encryption of a file before it is uploaded to the server

|  |  |
| --- | --- |
| **Scenario** | A user wants to upload a file, but the file is encrypted on the local device before being transferred to the server |
| **Triggering event** | User attempts to upload a file.  Backup timer attempts to upload a file. |
| **Actors** | User |
| **Related use cases** | Upload |
| **Stakeholders** | User |
| **Pre-**  **condition** | File needs to present on local filesystem.  Encryption key needs to be available. |
| **Post-**  **condition** | Encrypted file will be ready for upload |
| **Flow of**  **events** | |  |  | | --- | --- | | **Actor** | **System** | | 1. User attempts to upload a file | 1. Encryption key is loaded 2. File is encrypted and ready to upload | |
| **Exception** | Step Condition Action Description  2a. Encryption key not found Inform user that key doesn’t exist  2b. Key storage not found Set alarm and inform user that key storage is not connected  3a. Encryption fails Set alarm and inform user that encryption of file has failed. |

Decryption

Description – Decryption of a downloaded file

|  |  |
| --- | --- |
| **Scenario** | A file which has been retrieved is decrypted on local storage for use. |
| **Triggering event** | User attempts to access a downloaded file. |
| **Actors** | User |
| **Related use cases** | Download |
| **Stakeholders** | User |
| **Pre-**  **condition** | File needs to be fully downloaded |
| **Post-**  **condition** | Downloaded file will be decrypted and accessible to User’s operating system. |
| **Flow of**  **events** | |  |  | | --- | --- | | **Actor** | **System** | | 1. User attempts to access file | 1. Locates encryption key associated with file 2. Decrypts file with appropriate encryption key | |
| **Exception** | Step Condition Action Description  2a. Encryption key not found Set alarm and inform user that encryption key does not exist.  2b. Key storage not found Set alarm and inform user that encryption key storage is not connected.  3a. Decryption fails Set an alarm and inform user that decryption has failed. |

## Functional Requirements

Manage Servers

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Name | Requirement Description |
| FFR\_001 | Check Existence | The application shall check if the server exists before it can be added to server pool. |
| FFR\_002 | Enforce Uniqueness | The application shall prevent any servers from having the same name. |
| FFR\_003 | Server Details | The application shall provide a form for the user to enter server information. |
| FFR\_004 | Decommission | The application shall remove any backups that are currently on the selected server. |

Manage Encryption Keys

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Name | Requirement Description |
| FFR\_005 | Key Storage | Every encryption key shall be encrypted and stored on the available filesystem when not in use. |
| FFR\_006 | Key Length | Every encryption key shall be of 256bit length. |
| FFR\_007 | Key Compatibility | Every Encryption key shall be compatible with AES-256 encryption standard. |
| FFR\_008 | Key Generation | Keys shall be created by using a hash function which takes a user’s password and a randomly generated sequence of characters as inputs, in order to maximize entropy. |
| FFR\_009 | Commit Key Removal | There shall be a dialog box which prompts the user to confirm the deletion of an encryption key before removing the encryption key from the filesystem. |

Encryption and Decryption

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Name | Requirement Description |
| FFR\_010 | Encryption Algorithm | The encryption algorithm used shall be the Advanced Encryption Standard (AES). |
| FFR\_011 | Key Requirement | The encryption algorithm used shall require 256-bit encryption keys. |
| FFR\_012 | Key Uniqueness | Files shall only ever be decrypted with the same encryption key that was used to initially encrypt the file. |
| FFR\_013 | Key Selection | The decryption algorithm shall select the encryption key used for the encryption of the file if the encryption key is available. |
| FFR\_014 | Accessibility Alert | The decryption feature shall alert the user if the encryption key is not available for a downloaded file. |

Manage Backups

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Name | Requirement Description |
| FFR\_015 | File Selection | User shall choose the files he wants backed up. |
| FFR\_016 | Pool Selection | User shall choose server pool where the files are to be backed up. |
| FFR\_017 | Backup Details | User shall choose backup type and time. |
| FFR\_018 | Backup Removal | User shall choose which backup is to be removed from server pool. |

File Browser

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Name | Requirement Description |
| FFR\_019 | Hardware Availability | The user must have access to the hard drive we are requesting from |
| FFR\_020 | Filesystem Permission | The user must be able to view the files from the local client |
| FFR\_021 | Server Availability | The user must have access to the server we are requesting from |
| FFR\_022 | Server Filesystem Permissions | The user must be able to view the files from the remote server |

File Upload

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Name | Requirement Description |
| FFR\_023 | Server Storage | The server must contain enough room to hold the selected file(s) |
| FFR\_024 | Name Uniqueness | No two files can be named the same name |
| FFR\_025 | File Permissions | The user must have permission to move/delete files |

File Download

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Name | Requirement Description |
| FFR\_026 | Key Provisioning | An encryption key must be provided and match the user's account. |
| FFR\_027 | File Availability | A file or files must exist. |

File Delete

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirements Name | Requirement Description |
| FFR\_028 | Key Provisioning | An encryption key must be provided and match the user's account. |
| FFR\_029 | File Availability | A file or files must exist. |

File Replication

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Name | Requirement Description |
| FFR\_30 | Server List | A list shall be generated containing each server specified as a destination |
| FFR\_31 | Destination Accessibility | Each server specified as a destination shall be pinged to ensure that it is reachable before beginning the replication process |
| FFR\_32 | Storage Availability | Each server shall be queried for available storage space to ensure that there is enough space on the server’s file system for the files and/or directories to be stored before beginning the replication process. |
| FFR\_33 | Parallelism | A socket connection shall be opened to each server specified as a destination |
| FFR\_34 | Encryption Prerequisite | Files shall be encrypted prior to upload |

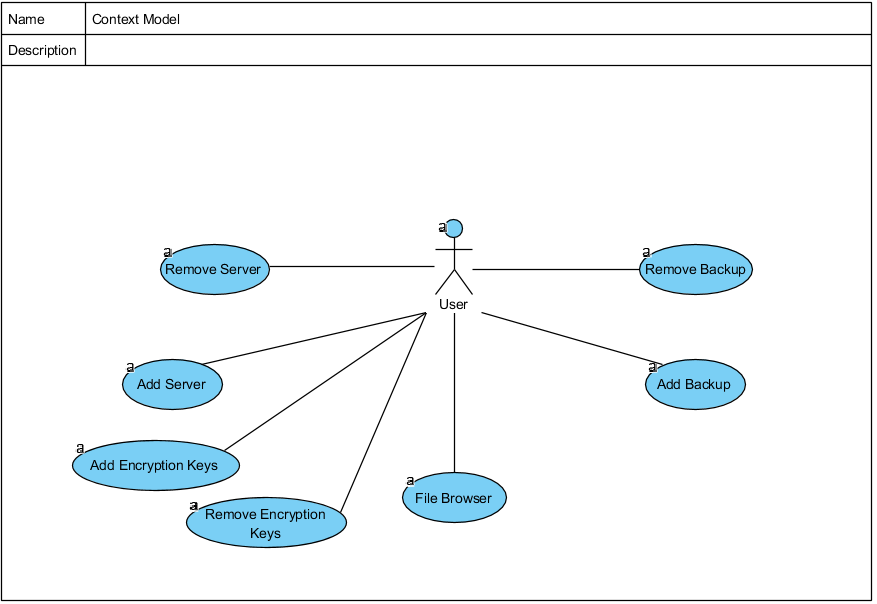
## Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| Requirement ID | Requirement Name | Requirement Description |
| NFR\_01 | Bandwidth | Any upload or download operation shall not utilize more than 90% of the available bandwidth of the computer running RCrypt |
| NFR\_02 | Software Language | All software shall be written in Java. |
| NFR\_03 | Scheduling Constraints | Only one upload or download operation shall occur at a time. In the event that the user attempts to start multiple upload/download operations, the user shall be notified that an operation is already in progress. In the event that a scheduled backup attempts to start one or multiple operations, the backup shall be delayed until the current operation completes. |
| NFR\_04 | Hardware Constraints | The users CPU will determine the speed of RxCrypt application and in turn the users experience. The application will need to be lightweight in order to anticipate any users that may have weak CPUs. |

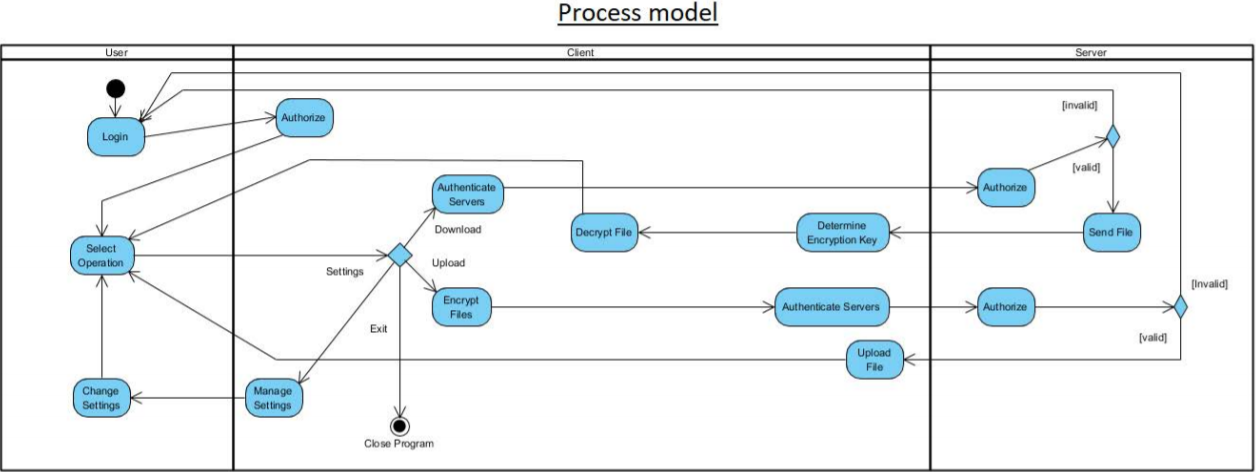
# Design

## System Modeling

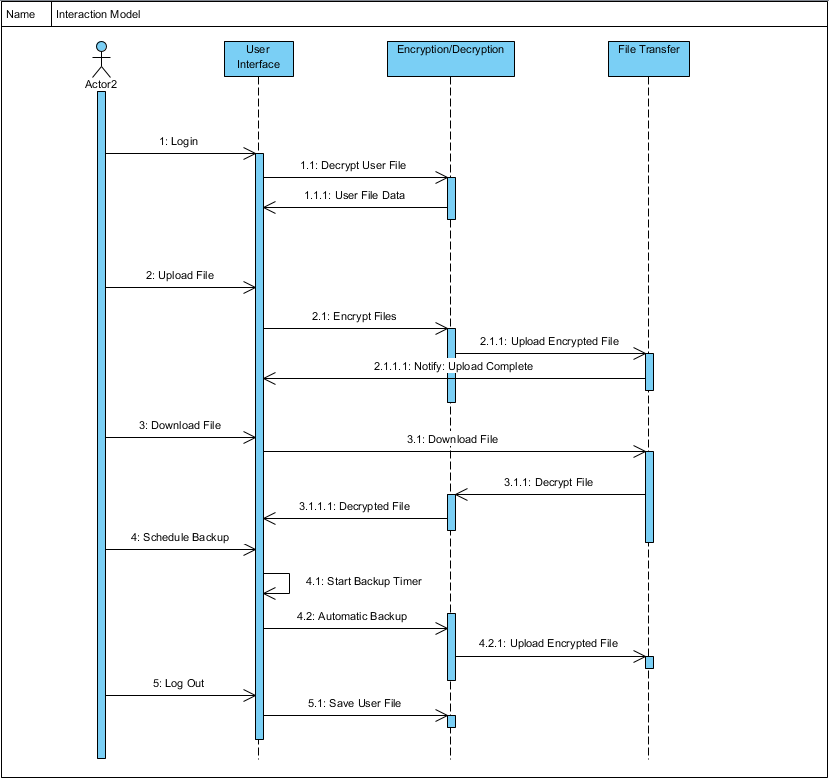
### Context Modeling



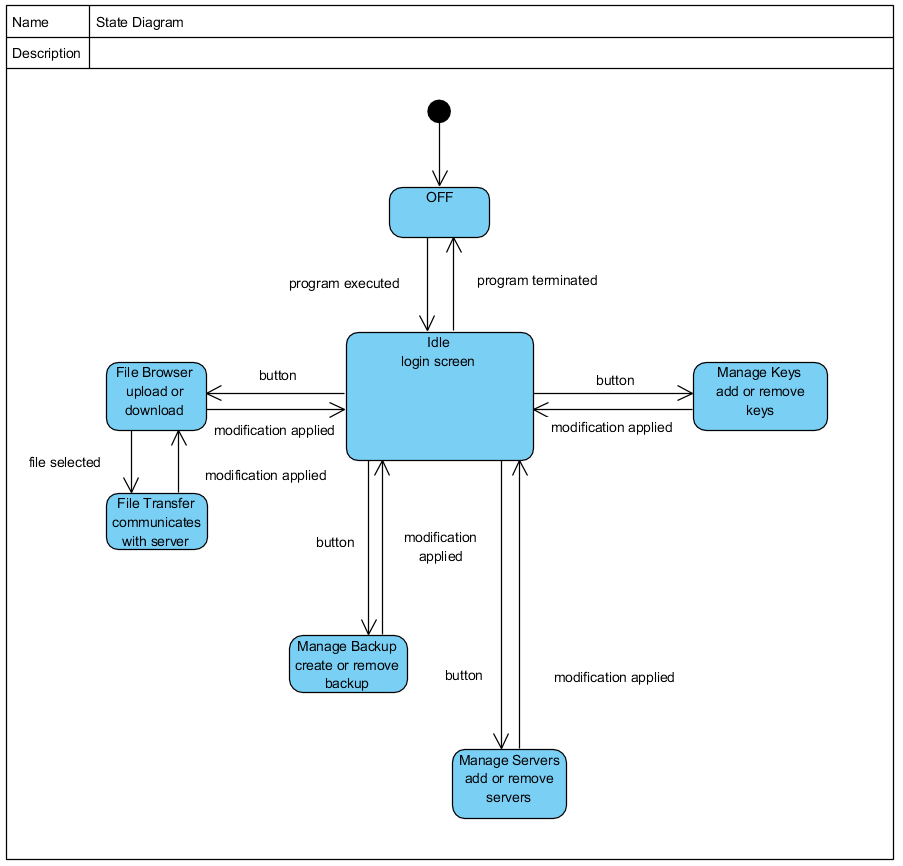
### Process Modeling



### Interaction model

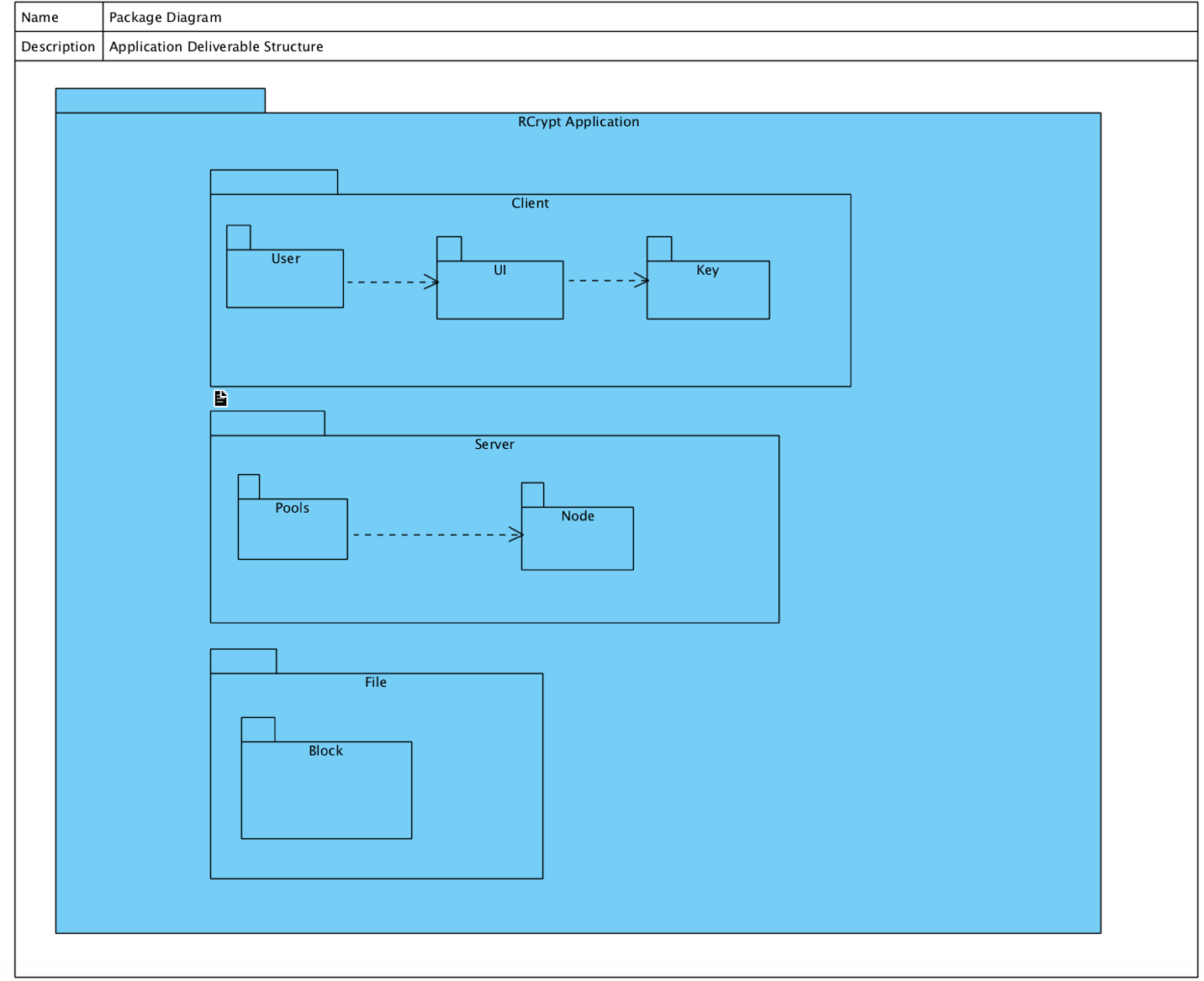


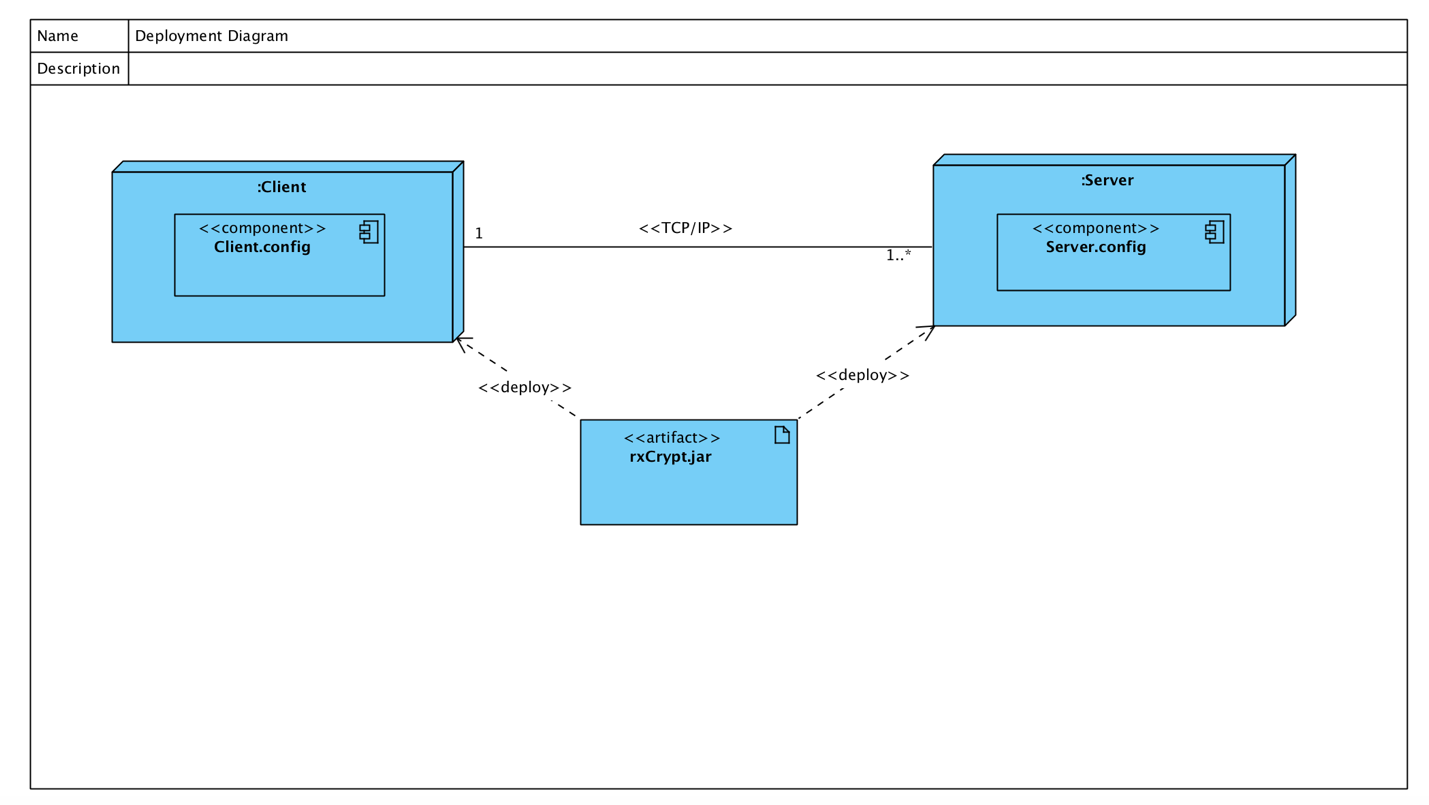
### Behavioral Modeling



### Structure Modeling

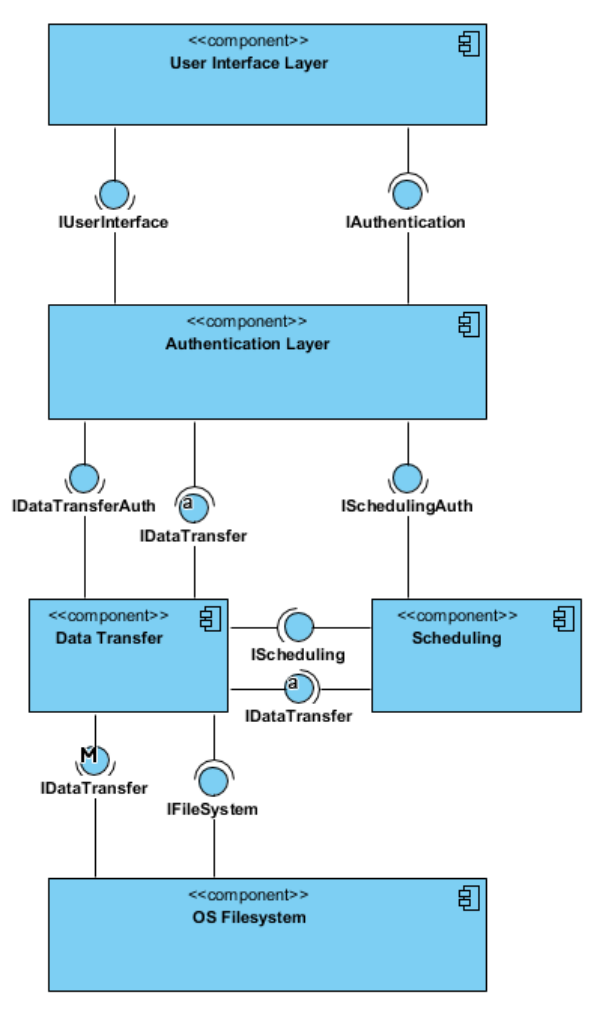
### 





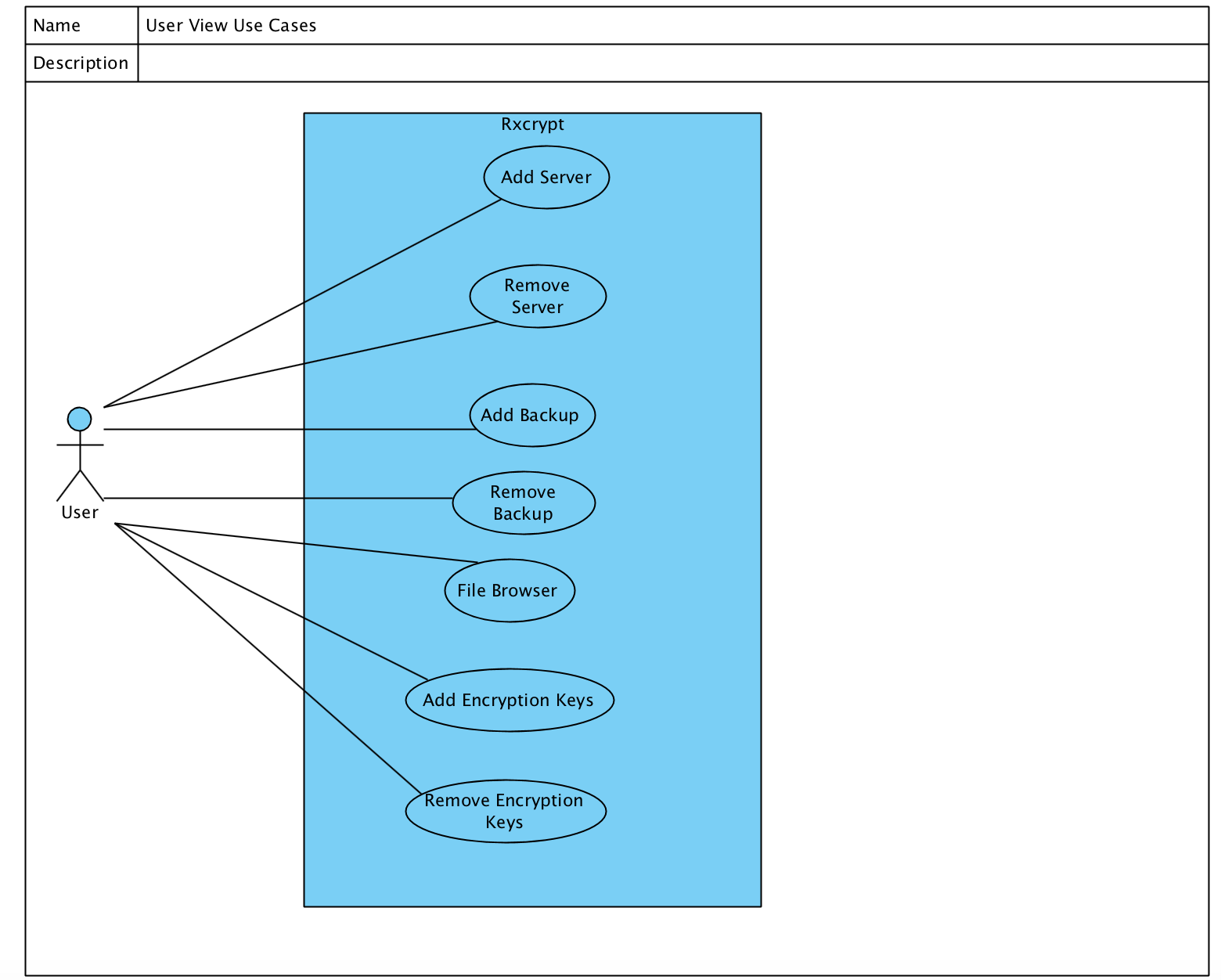
## System Architecture and Patterns

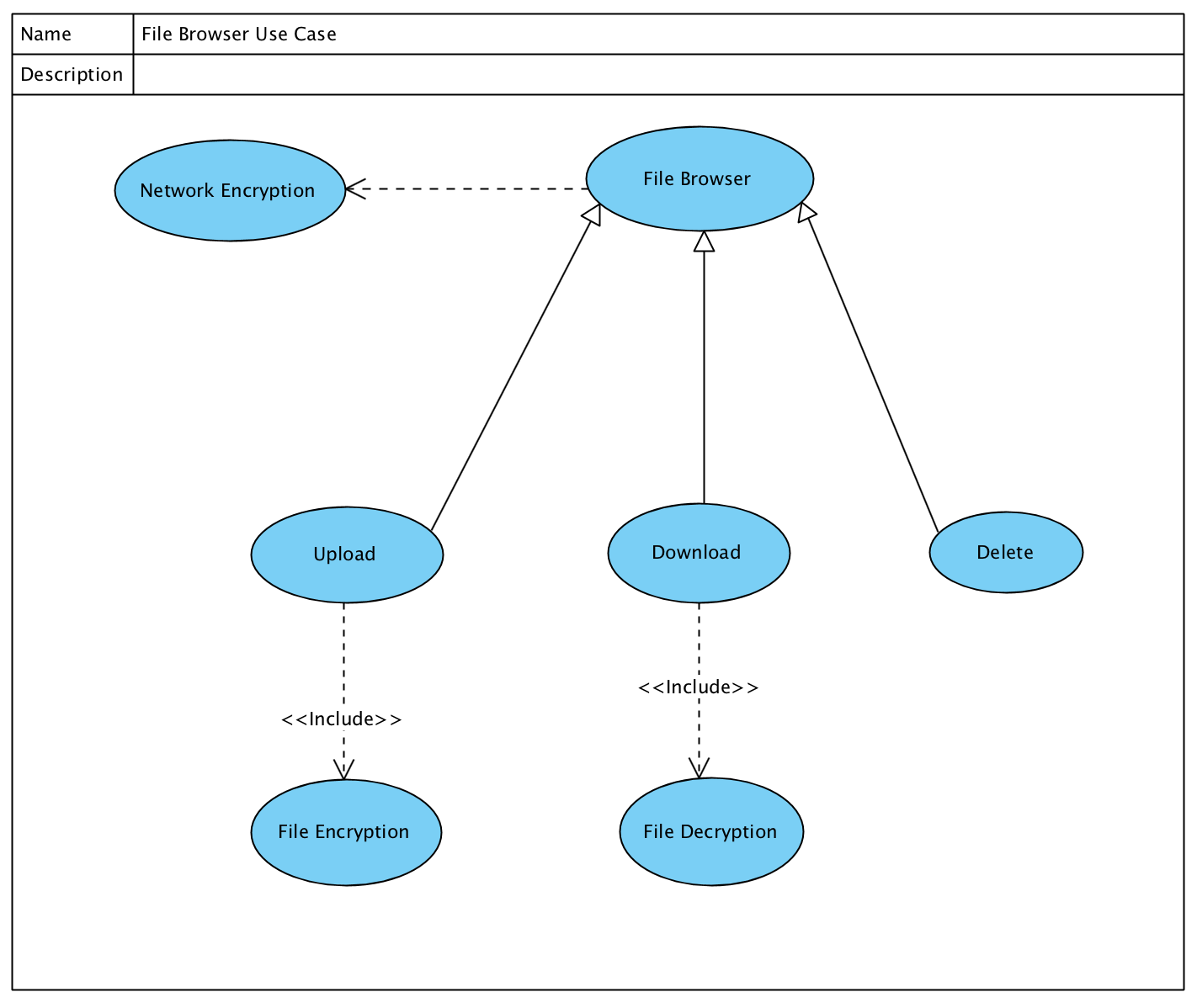
We decided to use a layered architecture pattern to allow each group member to individually develop their portion of the overall project and combine each layer together to create the final project. This architecture proved to be immensely beneficial as different layers of the project were developed in parallel. Without this boon to productivity, this project would not have been completed by the deadline.



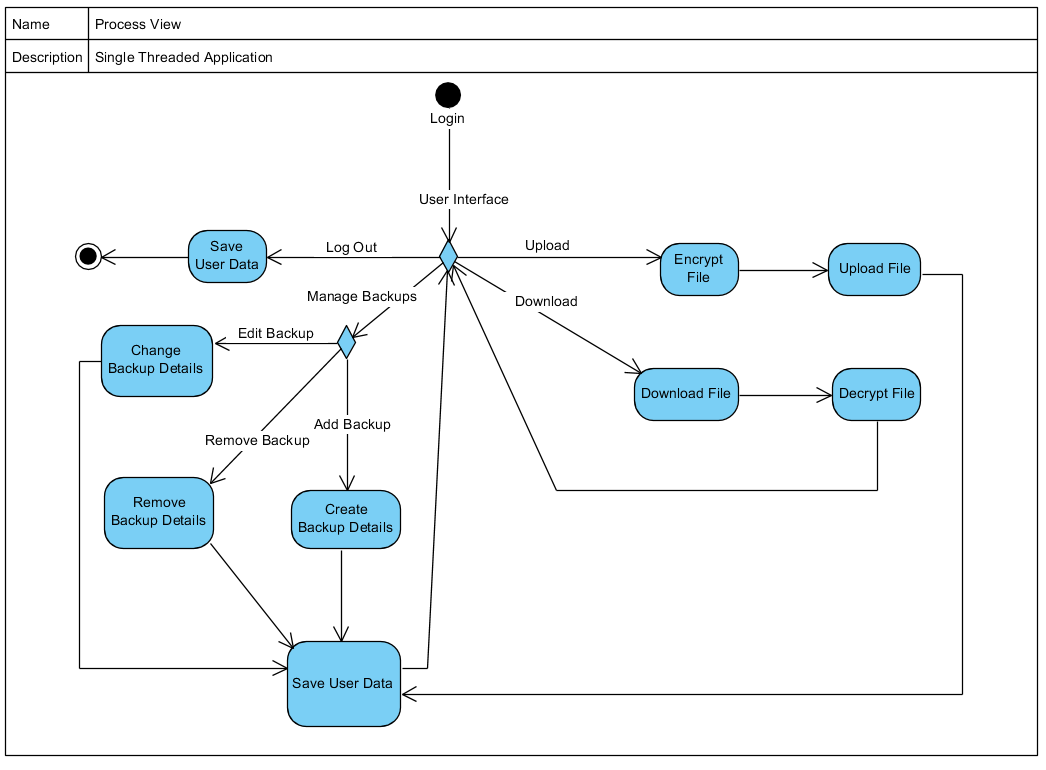
## 4+1 views

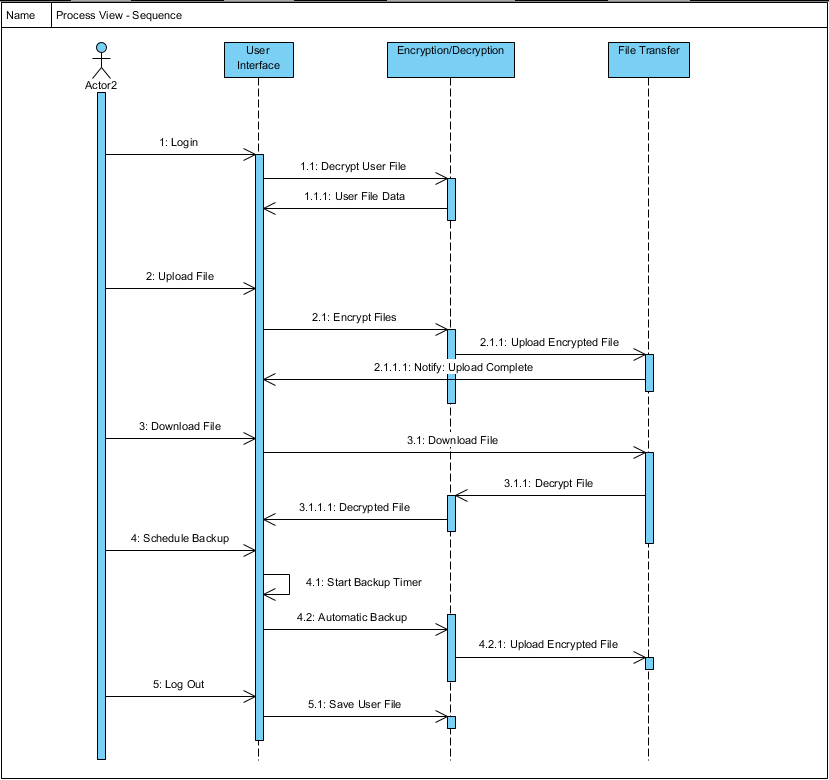
### User View



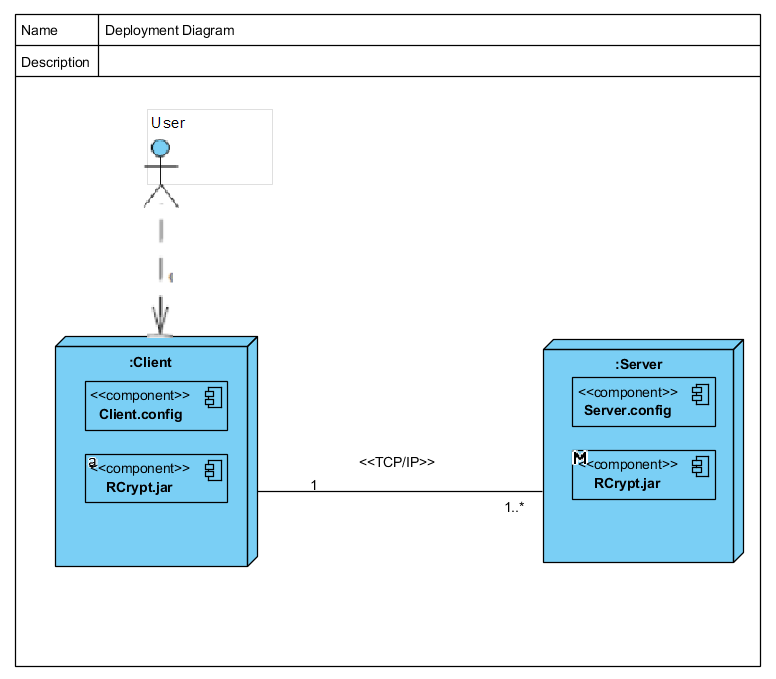


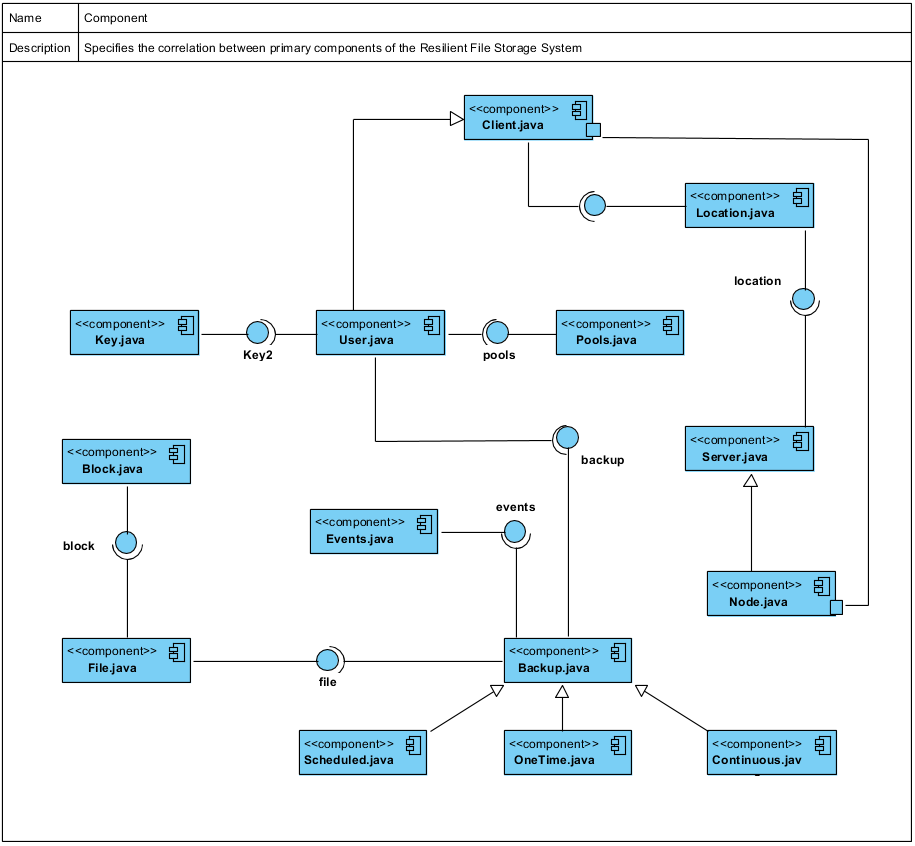
### Process View



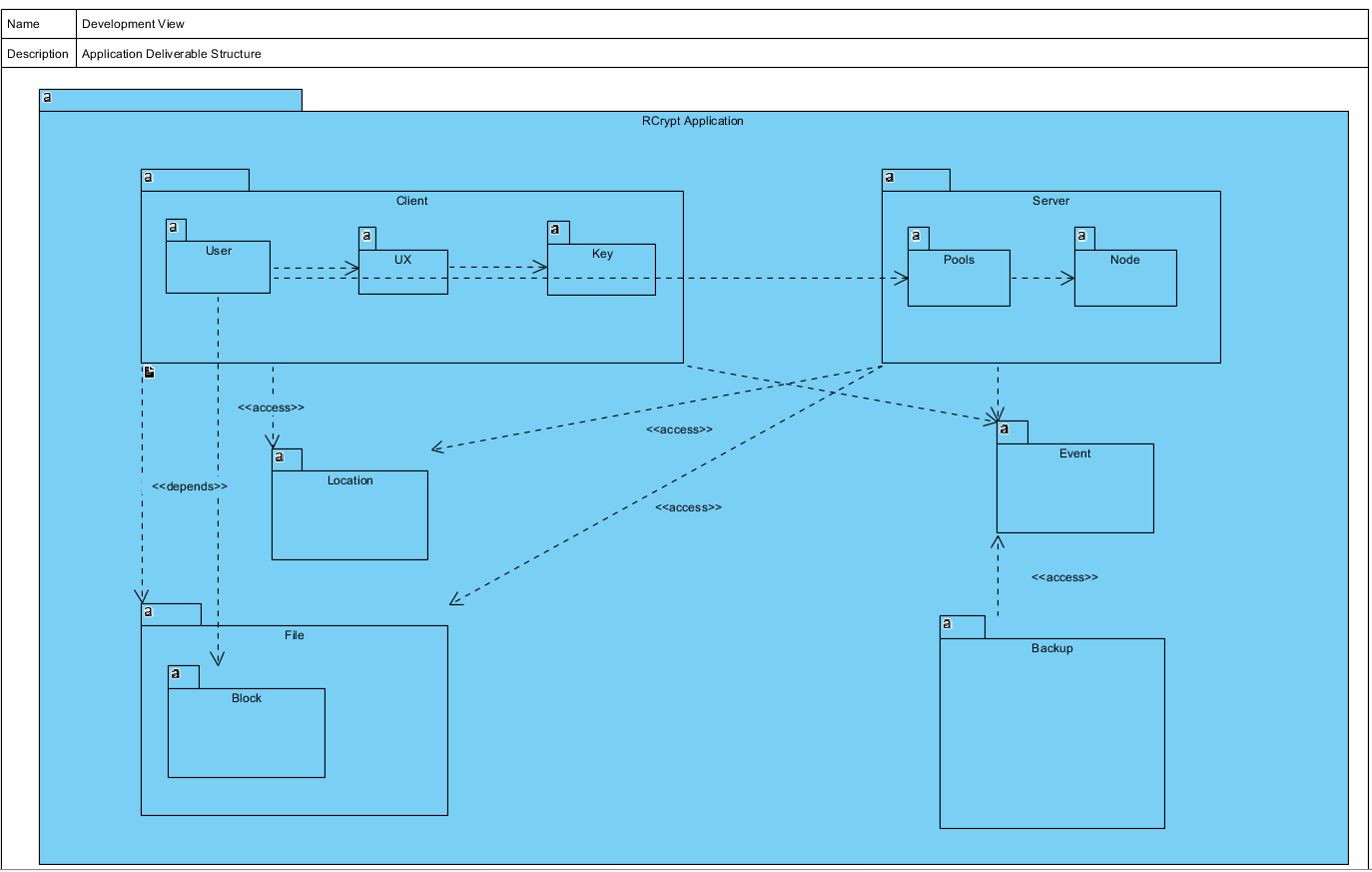


### Physical View

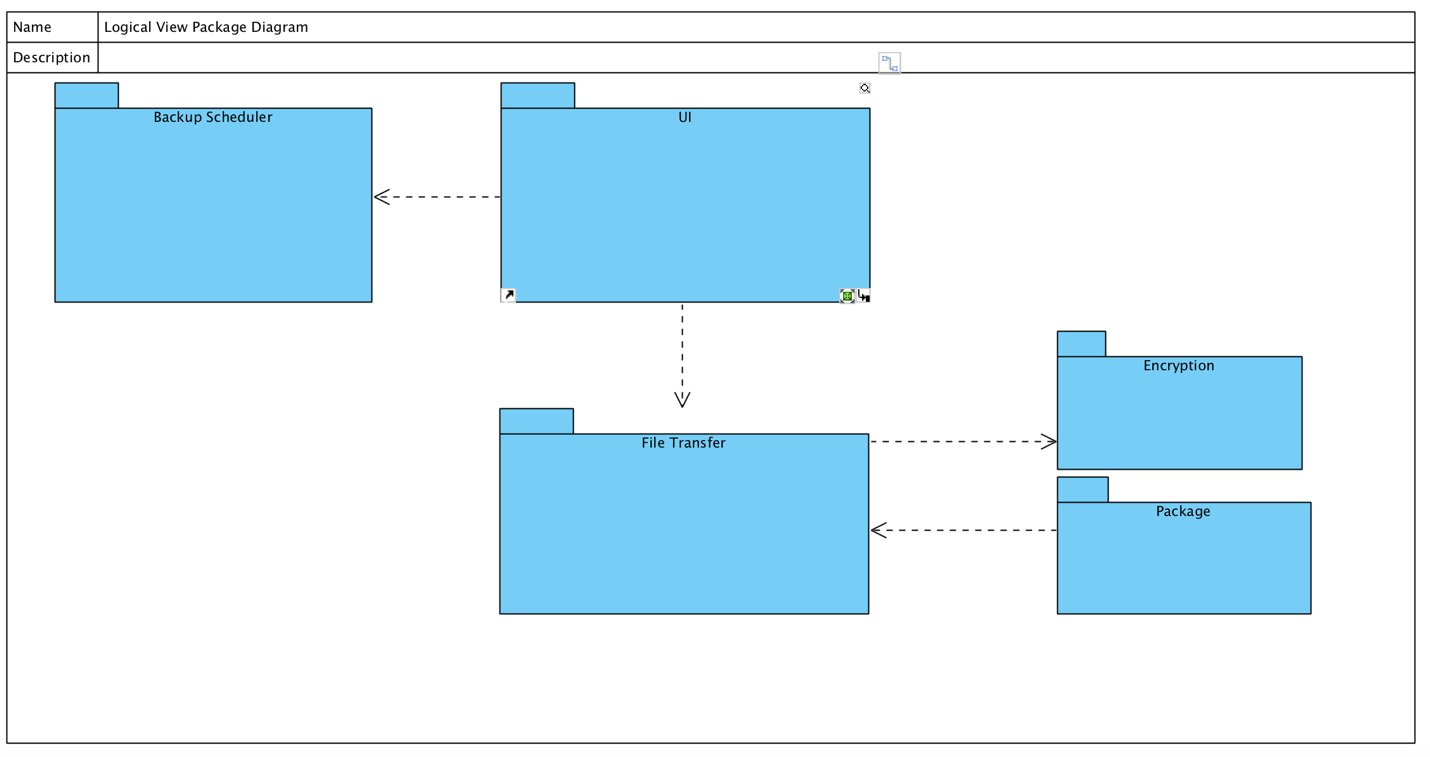




### Development View



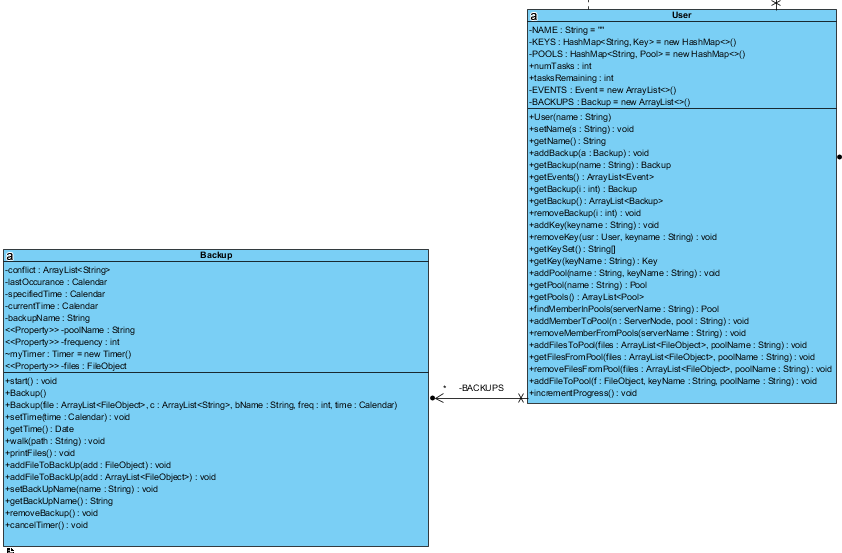
### Logical View



## 

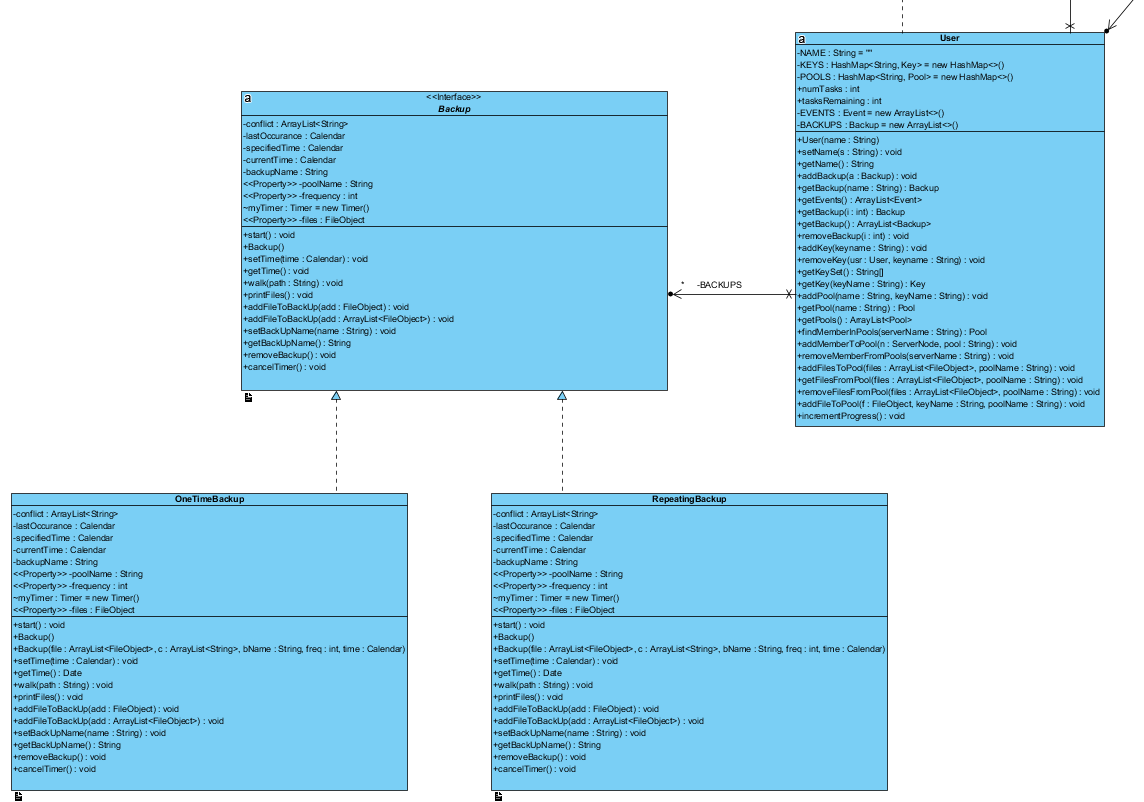
## Detailed Design Principles

Because of the direct and singular nature of our software, there are very few opportunities to exercise the design principles. One opportunity for examination, however, is with the “Backup” class violating the OCP design principle. There are two types of backups that are handled in this class. First, repeating backups are ones which occur at certain intervals. Second, one-time backups are simply the backup operation being scheduled to occur once at a later date. The original implementation is described in a segment of the class diagram below:



*Violates OCP*

To remedy this, the backup class is converted to an interface and is realized by the segment of the class diagram below. The two types of backups realize the backup interface class, and so they can be individually altered without affecting the interface class. Hence the other class is not affected by the change.



*Adheres to OCP*

# Testing

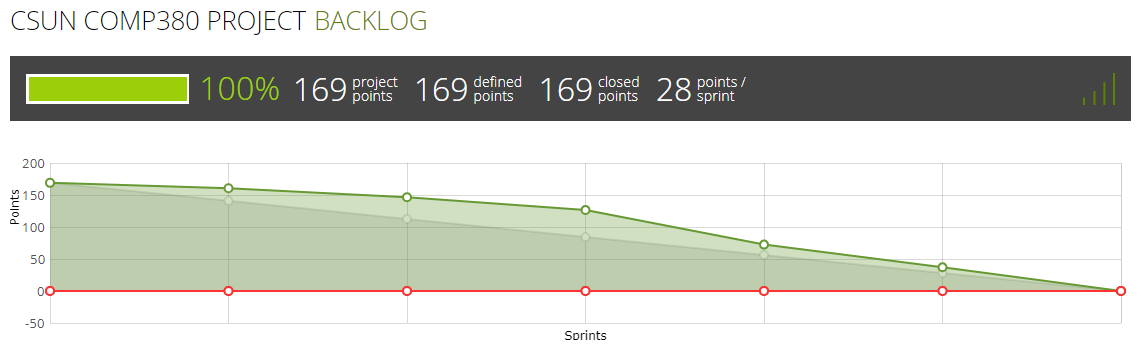
Walkthrough and inspection testing were performed during development and after the functional prototype was released. Several small issues with edge cases were remedied during this phase. Unfortunately, there was not enough time to perform comprehensive black box testing of the software product. We were able to perform basic execution testing prior to presenting our software product. This included verifying that all features functioned as intended and validating operational parameters of the features. This has led to a stable software product which was acceptable for demonstration purposes.

# Project/Process

## Open Issues

Unfortunately we did not have enough time to thoroughly test and optimize our code. If we were to continue to pursue developing this software we would perform more walkthrough and inspection testing as well as white and black box testing. In addition, our UI is purely functional and going forward could use improvements in design, potentially even requiring the help of a graphical artist.

## Retrospective

We had chosen to use SCRUM as a project management model for this project for two reasons: First, it is a popular choice and learning to work with SCRUM in a college environment is ideal. Second, because each member has a busy schedule, it was believed that an agile model would be best suited for a disparate team. However, the single most import function of SCRUM are the daily meetings, and this has been made very clear during this project. Without meeting regularly and frequently the development of the project was slow. This can be seen in the burndown chart below.

We had used taiga.io as our SCRUM project management tool, and while it doesn’t show the dates of the sprints in the burndown chart, it does show how the project fell behind schedule. The faint grey line hidden within the green is the ideal burndown line, and the mad rush to finish the product by the deadline can be seen in the last three sprints. Additionally, because our requirements did not change frequently, several other benefits to the agile model were lost on this project. Nonetheless, iteratively producing a working product did improve both productivity and quality.