NYU | TANDON SCHOOL OF ENGINEERING

## Computer Science and Engineering

# *Library Tracker* System Requirements and Analysis Specification (RAS)

**(Version 1.3)**

Document Number: RAS-001

| | |
|---|---|
| Project Team Number: | B27 |
| Project Team Members: | Girish Ramloul (gr1188)<br>Ricardo Gutierrez-alvarez (rga267)<br>Viola Rreza (vr714) |

## REVIEW AND APPROVALS

| Printed Name and Title | Function | Date | Signature |
|---|---|---|---|
| Mr. Girish Ramloul | Author | 2/13/19 | GR |
| Mr. Ricardo Gutierrez | Author | 2/13/19 | RG |
| Miss Viola Rreza | Author | 2/13/19 | VR |

## REVISION LEVEL

| Date | Revision Number | Purpose |
|---|---|---|
| 10/04/18 | Version 1.0 | Initial Release |
| 10/19/18 | Version 1.1 | Revised RAS |
| 11/20/18 | Version 1.2 | Analysis Added |
| 2/13/19 | Version 1.3 | Revised RAS |

## Table of Contents

# 1. INTRODUCTION

## 1.1 Purpose

The purpose of the initial release of the RAS is to define the business domain and certain sections of the document excluding the requirement and analysis section. The audience for this document is the software quality group, the development group, the requirements group, and project management group. And any other group that might be interested in reading this document.

# 2. SCOPE

The system that will be produced is a mobile application that will keep track of all items a student has checked out of the Bern Dibner Library as well as what times the items are due. Let's say a student forgot what time an item is due and loses their receipt, the only current way they can tell when their items are due is by logging online through their NYU account and going through the library system to find the items they have checked out or by inquiring at the front desk, which requires its own extended process. Most of the time, both of these processes are slow and inconvenient and it just leads to students not returning their items on time and receiving late fee charges that can go up to $250 depending on how late it is. Students lack a convenient way to check their return times and also lack the ability to see what is available and if they have a fine. In our mobile app you will be able to login using your NYU credentials and in easy and convenient way be able to see and interact with what items you have checked out and when they are due. In addition, the app will have the ability to send you push notifications when your items are approaching their due times. The student should also be able to check his/her balance in addition to being able to pay it off. The application will also allow students the ability to see what certain items are available. Lastly, the application will grant a student the ability to reserve an item for up to 45 minutes before it cancels their hold on the item.

## 2.1 Identification

This is the Requirements and Analysis Document, the business specification. This is version 1.0. Project team name is A27. Project name is Library Tracker. Date is 10/04/18.

## 2.2 Bounds

We will design and implement an API to interact with the Libraries database system so that we can retrieve what items are checked out under a
certain NYU students account. The client or user of the system will also input their own NYU login credentials

## 2.3 Objectives

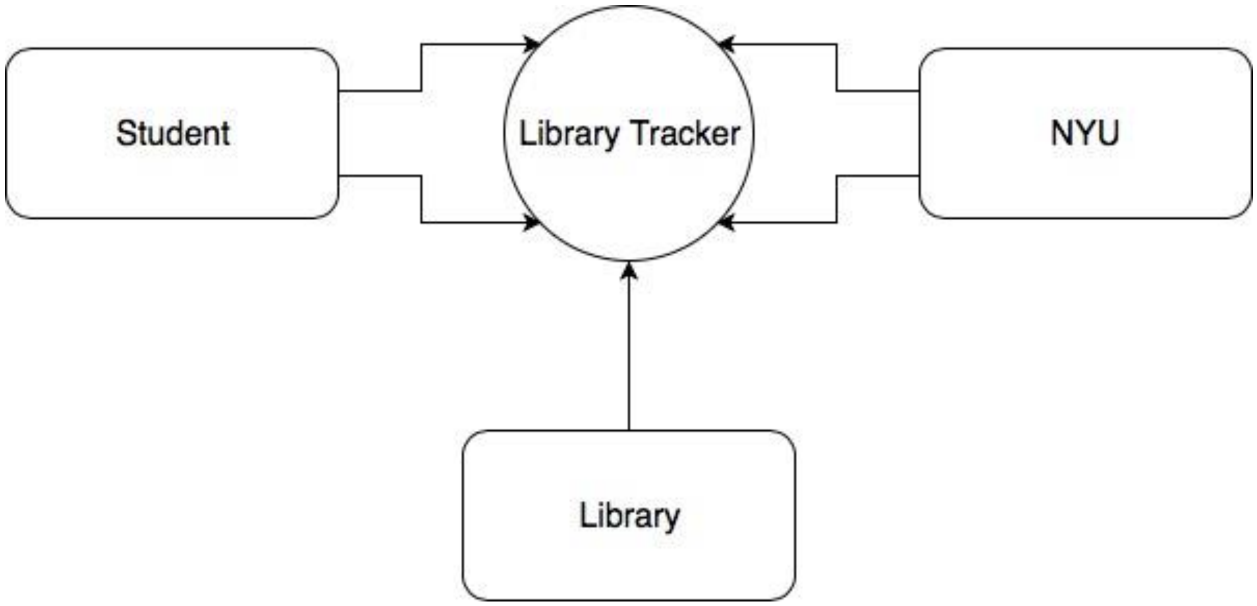
The renters are the end-users of most of the features in the application and since they are mostly students, the team does not anticipate rigid deadlines on the project. However, to ensure the application meets the requirements and quality metrics, the project manager requires the team to deliver 25% of the work within the first month. We will be using an incremental life cycle for this project.

| Project Deliverables | Date Due |
|---|---|
| Project Proposal | 2/13 |
| Software Requirements and Analysis Documentation (RAS) | 2/13 |
| Software Project Management Plan (SPMP) | 2/27 |
| Software Design Document (SDD)- Initial | 3/6 |
| Software Design Document Final | 4/26 |
| Implementation/Demonstration | 4/26 |
| Final Presentation | TBD |

# 3. OVERALL SYSTEM OVERVIEW

## 3.1 Context Diagram

The application is dependent on a library’s database along with the scanning equipment used at the library’s front desk



## 3.2 Additional Descriptive Items

a) The application must serve as a way to remind us to turn in an item before it becomes overdue. It should send a simple push notification to a student's phone a certain number of hours before the late fees begin. As soon as the student checks out the item and said item is scanned, the application is automatically updated. The student can update his/her settings to decide how many notifications they would like and how many hours/minutes in advance they would like them sent in. Within the application as well, we would like to see the availability of certain items. We also feel that it would be beneficial to let students reserve items ahead of time with a time constraint. The student should also be able to check his/her balance in addition to being able to pay it off. Like most cash transfer applications, the application should be able to accept credit card information and allow the student to transfer funds to Dibner in order to pay off any fines.

b) Users of this app include students who want to avoid the possibility of not turning in an item into the library on time. They are generally tech savvy and will understand how to log in using their NYU account and customize notifications.

c) Library employees will also use the app as well to update the availability list and facilitate the checking out of items for students.

d) The team will need to devise a way to secure money transactions over the platform. The lack of an IT risk analyst in the team makes the task more challenging.
e) The application has to be compatible with both iOS and Android systems. Some of the features may be different depending on the operating system, which can add delays.
f) The application will be tested with large sets of data prior to deployment but some of the limitations will only show up after the project is live. The team has the responsibility to reduce bugs to prevent the system to crash when it is implemented.
g) The team intends to use open-source resources but the specific features of the application may require tools that are not easily accessible. The team will have to adapt while making sure quality is not compromised.
h) Permission is needed from NYU in order to be able to use NYU Dibner's database. If permission is not granted, a unique database will be created in order to showcase the functionality of the application. This unique database will mimic that of a working library. Permission is also needed in regards to allowing students to be able to pay off late fees through an application rather than in person.
i) It is assumed that the first iteration of the application will be done so using a database that mimics that of NYU Dibner Library. If NYU grants permission to its databases, it will allow a future iteration to actually include the items and student profiles in Dibner's database.

## 4. DOCUMENT OVERVIEW

Section 2 introduced the scope of the project. The overall system overview can be found in section 3. The rest of the Version 1.0 contained the business requirements, system test plan requirements, qualification provisions and requirements traceability. The revised version (1.1) also includes the functional requirements. The status of the RAS can be traced from the section "Evolution of the RAS", found at the end of the document.

# 5. REFERENCE DOCUMENTS

Project Proposal Library Tracker, Team A27, Version 1.0, September 20, 2018
RAS Business Requirements, Team A27, Version 1.0, October 04, 2018

# 6. BUSINESS REQUIREMENTS

## 6.1 Technology

The application will render the process of borrowing books more efficient. Unlike the current system in place, it can provide updated information on the availability of books. It is also a platform for renters to pay any due fine securely.

## 6.2 Economics

The application will reduce the number of staff required for assistance, thereby reducing labor cost.

## 6.3 Market Considerations

The demographic will mostly be college students.

## 6.4 Risks and Alternatives

The following tables list possible system risks and alternative solutions for each risk:

| | |
|---|---|
| **Business Risk** | Deadline Overrun |
| **Probability** | 50% |
| **How discovered** | Project Schedule |
| **Responsible Party** | Team A27 |
| **Status** | Required |

| | |
|---|---|
| **Mitigation Party** | Assume/Accept |

| | |
|---|---|
| **Operational Risk** | Data Security |
| **Probability** | 50% |
| **How discovered** | Simulations |
| **Responsible Party** | Team A27 |
| **Status** | Required |
| **Mitigation Party** | Avoid |

| | |
|---|---|
| **Technology Risk** | Organizational Change |
| **Probability** | 25% |
| **How discovered** | Performance and Technical Specifications |
| **Responsible Party** | Team A27 |
| **Status** | Required |
| **Mitigation Party** | Watch/Monitor |

| | |
|---|---|
| **Economic Risk** | Budget Overrun |
| **Probability** | 50% |
| **How discovered** | Project Cost Estimate |
| **Responsible Party** | Team A27 |

| | |
|---|---|
| **Status** | Required |
| **Mitigation Party** | Assume/Accept |

## 6.5 Human Resources and Training

The project will require both front and back end development expertise. The team only has prior experience in database management and adequate web design. To build the application, the team will also have to be versed in User Interface design and Mobile app development.

# 7. SPECIFIC REQUIREMENTS (DESCRIPTIVE FUNCTIONAL REQUIREMENTS)

## 7.1 Functional Descriptive Detailed Requirements

Four functional requirement definitions have been identified for the system, namely: user access, user interface, security and administrative.

1. The following functional requirements describe the system's **user access** functionality:
   a) The system shall grant access to an existing user after user provides correct username and matching password.
   b) The system shall create a new user if user wishes to sign up. The database is updated accordingly.
   c) The system shall generate a pop-up message if wrong credentials are keyed in. A link will be provided to reset the password.

2. The following functional requirements outline the system's **user interface** functionality:
   a) The system shall prompt user for log in details using text boxes styled using CSS forms.
   b) The system shall allow user to navigate smoothly to the following links: book rental history, amount of fine, fine due, books past due and time left/time past due, payment and books availability.

3. The following functional requirements describe the system's **security** functionality:

a) The system shall authenticate user login details using the OAuth protocol/framework.
b) The system shall provide a safe link for user to pay using the Braintree payment gateway.

4. The following functional requirements outline the system's **administrative** functionality:
a) The system shall give unlimited access to admin. Admin can delete a user or flag a user. The admin updates the database.
b) The system shall generate the monthly reports on the rental history for that month and the fines due with the students' names. Reports are based on the database and generated in the XML format for Excel view.

## 7.2 Requirement Use Cases

The system's user access functionality will allow the student to do the following:

1. **Sign up**:
The system shall allow new users to sign up using their school email which will result in them being added to the database. In order to successfully sign up, the user must verify their account through an email sent by the system.

2. **Login:**
The system shall allow the user to log into the system using their correct credentials. If the user inputs an incorrect username or password, the system will warn the user that the username and/or password is incorrect.

The user interface will allow a student to do the following:

1. **Check time left to past due:**
The system shall let a student who is logged into the book rental system to track how much time is left before the rented book is past due. The user will be notified 30 minutes before the past due time. The user will also be notified when the rented book is past due.

2. **Check balance:**
The system shall allow the user to check how much fine has been accumulated. The system shall notify the user every 24 hour as a reminder.

3. **Pay fine:**
The system shall provide a payment gateway for user to pay any remaining fines. The gateway will accept any form of wire transfer authorized by the school library.

4. **Check availability:**
The system shall permit the user to look for books in stock and their availability. If none of the requested book is currently available, the user can request a notification which will be sent once the book is available based on first-come-first-served basis.

5. **Request rental extension/book:**
The system shall allow the user to extend a rental time, if the admin wishes. Books with the extendable feature will be marked differently. The system shall provide a form for the user to request a book that is currently not in stock.

The system's security functionality will allow for the following:

1. **About previous rentals:**
The system shall keep track of each user's rental history. The user can refer to the history page. The security of the system will make sure that the information remains correct.

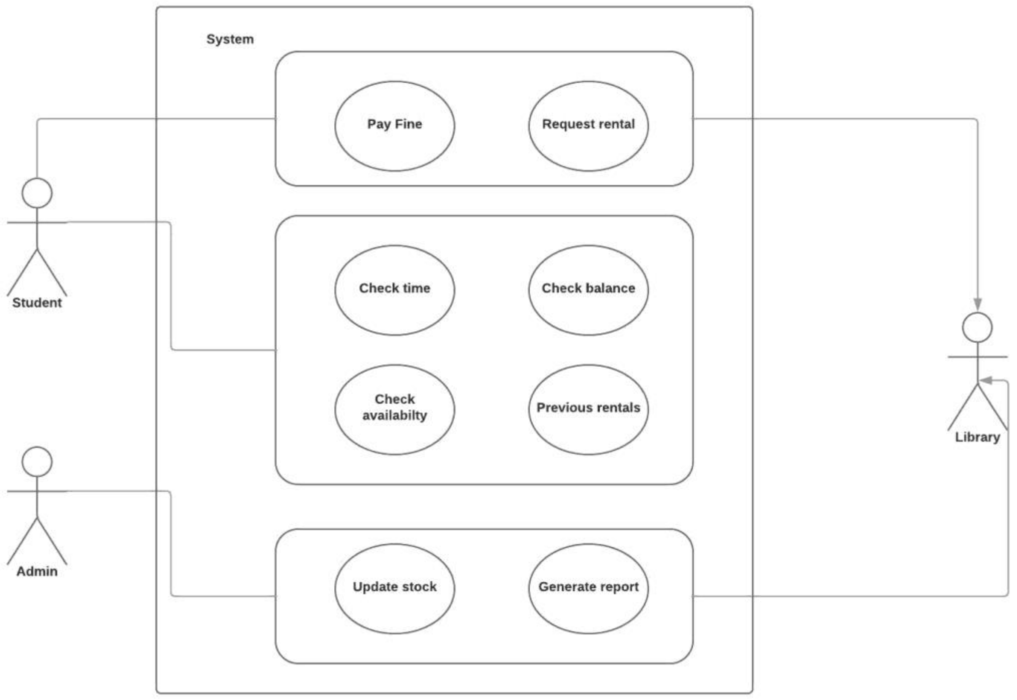
2. **Keep payment information safe:**
The system shall maintain the safety of the each user's payment information. Because the user may save credit card information in order to pay fines, the system's security is a priority.

The system's administrative functionality allows for the following:

1. **Display/Print reports:**
The system shall give the admin access to the monthly reports on the rental history for that month and the fines due with the students' names. The reports can be generated in spreadsheet or pdf, should admin decide to print the reports.

2. **Update stock:**
The system shall allow the admin to update the database when a new book is made available or a book is discontinued.

## 7.3 Use Case Diagrams



## Use Case Descriptions

| Check time left to past due | | |
|---|---|---|
| **Description** | Time left for rental to be past due | |
| **Pre-Conditions** | Book rental | |
| **Flows** | **Basic or Normal Flows** | 1. Get rental time<br>2. Get time due |
| | **Alternative Flows** | |
| **Post Conditions** | Fine if past due | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

| **Check balance** | | |
|---|---|---|
| **Description** | Balance due | |
| **Pre-Conditions** | Book rental; fines; late returns | |
| **Flows** | **Basic or Normal Flows** | 1. Get rental history<br>2. Calculate total amount |
| | **Alternative Flows** | 1. Pay fine |
| **Post Conditions** | | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

| **Pay fine** | | |
|---|---|---|
| **Description** | Pay amount due through payment portal | |
| **Pre-Conditions** | Fines; late returns | |
| **Flows** | **Basic or Normal Flows** | 1. Payment portal |
| | **Alternative Flows** | |
| **Post Conditions** | Update balance | |
| **Special Requirements** | Payment gateway solution fee | |
| **Extension Points** | None | |

| **Check availability** | |
|---|---|
| **Description** | Books currently available |

| | | |
|---|---|---|
| **Pre-Conditions** | Books in stock; rental | |
| **Flows** | **Basic or Normal Flows** | 1. Get rental history<br>2. Get stock |
| | **Alternative Flows** | |
| **Post Conditions** | | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

| **Update stock** | | |
|---|---|---|
| **Description** | Add new book to database | |
| **Pre-Conditions** | New book | |
| **Flows** | **Basic or Normal Flows** | 1. Add book details |
| | **Alternative Flows** | |
| **Post Conditions** | Update database | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

| **Request rental extension/book** | |
|---|---|
| **Description** | Extend rental or request new book for stock |
| **Pre-Conditions** | Rental |

| | | |
|---|---|---|
| **Flows** | **Basic or Normal Flows** | 1. Send form to admin |
| | **Alternative Flows** | |
| **Post Conditions** | Admin approval | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

| **Display/Print reports** | | |
|---|---|---|
| **Description** | Monthly report on rental history | |
| **Pre-Conditions** | Rental | |
| **Flows** | **Basic or Normal Flows** | 1. Get rental history<br>2. Get money transaction |
| | **Alternative Flows** | |
| **Post Conditions** | Report Analysis | |
| **Special Requirements** | None | |
| **Extension Points** | None | |

| **About previous rentals** | | |
|---|---|---|
| **Description** | Rental history of user | |
| **Pre-Conditions** | Rental | |
| **Flows** | **Basic or Normal Flows** | 1. Get rental history |
| | **Alternative Flows** | |

| | |
|---|---|
| **Post Conditions** | |
| **Special Requirements** | None |
| **Extension Points** | None |

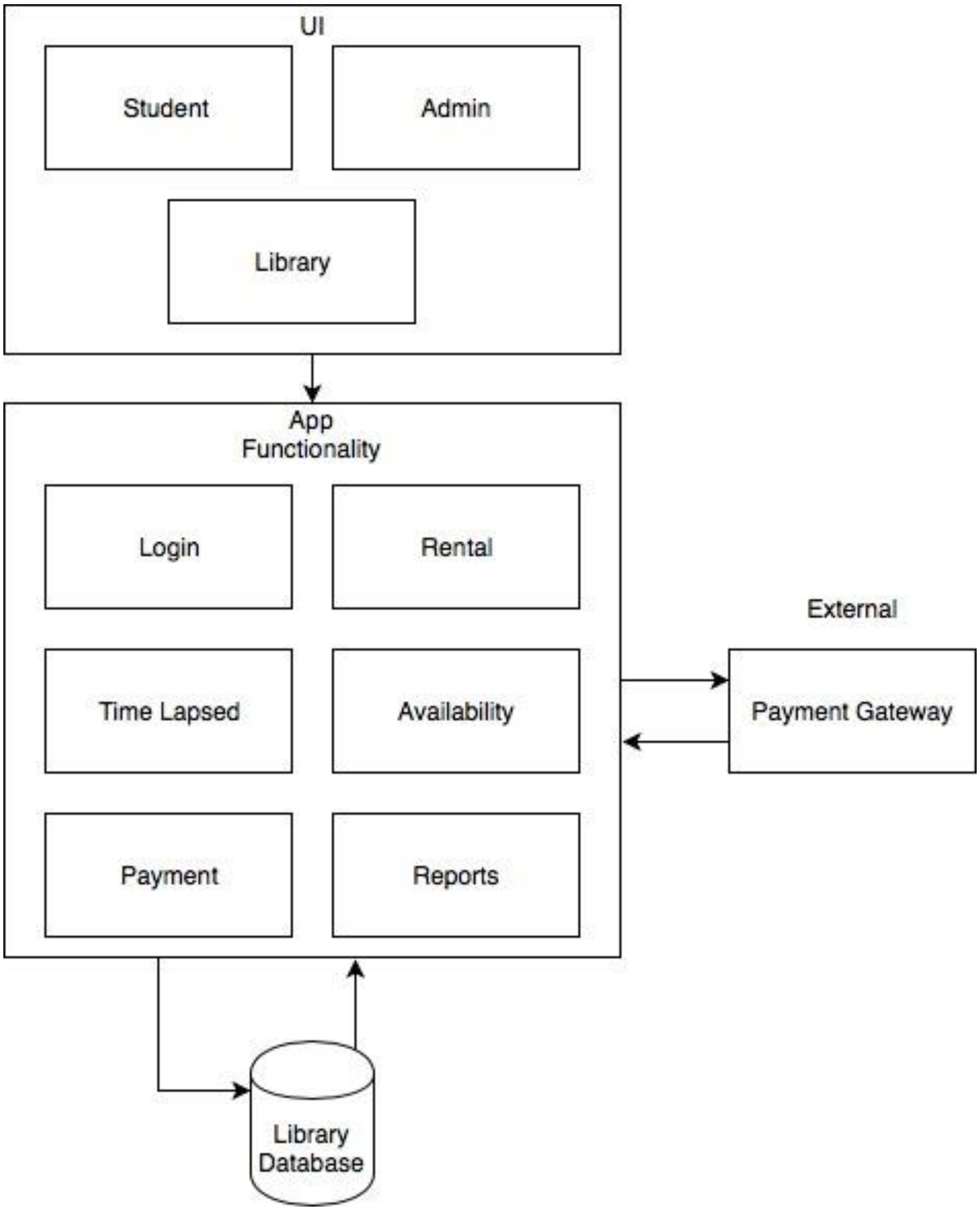
## 9. ANALYSIS

### 9.1 Component (Component/Package/Subsystem) Architecture

A three-tiered architecture will be used in this documentation to describe Library Tracker's architecture. The architectural diagram can ve found in section 9.2.

### 9.2 Component Architecture Diagram

The diagram below represents the Computer Architecture Diagram of the Library Tracer App.



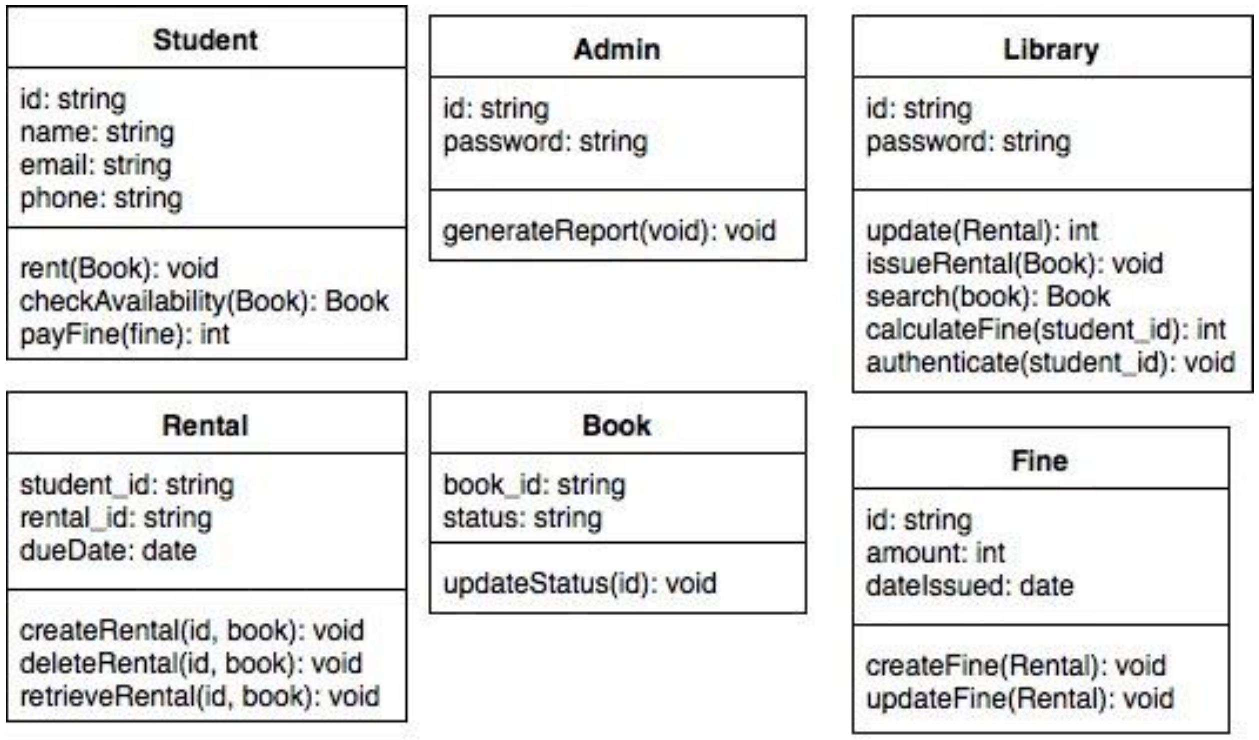
## 9.3 Component Descriptions

The system architecture has been broken down into four application layers: User Interface, App Functionality, Library Database and External. The UI layer consists of the three anticipated users of the app: student (end user), admin (request reports) and library (responsible for updating repository). The App Functionality layer consists of the main functional components of the app: user login with authentication, rental system, time lapsed from rental, availability of rental, payment option and report generation. The UI layer complements the App Functionality layer. The App Functionality layer makes periodic updates to the Library Database layer. For secure payment options, an API is set up to connect the Payment Gateway in the External layer to the app.

## 9.4 Class Diagrams

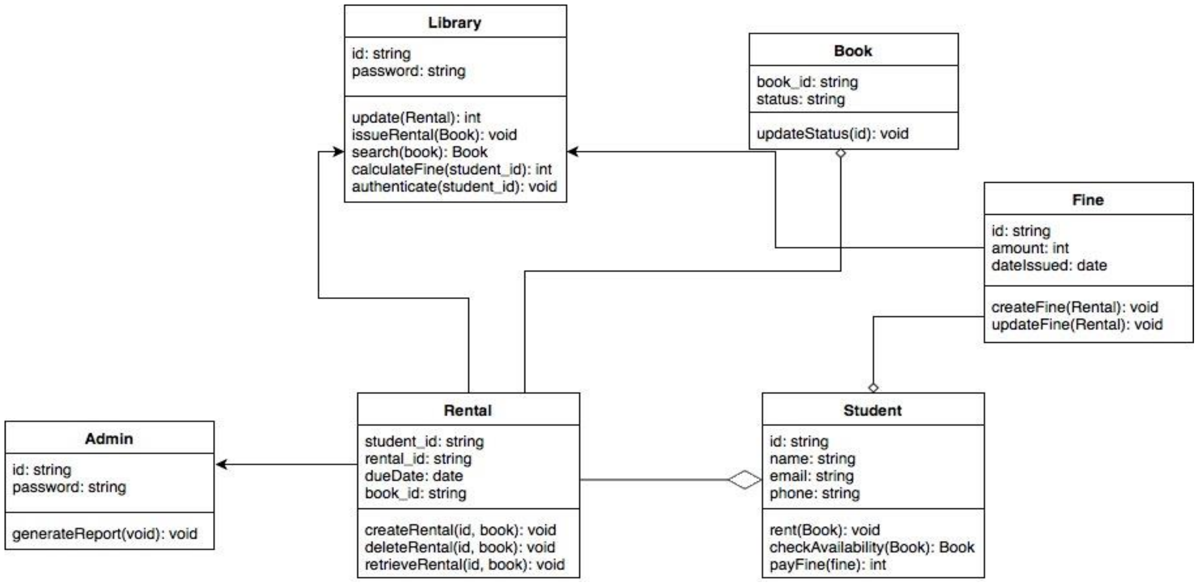
### 9.4.1 Individual Class Diagrams

The individual class diagrams are represented in the diagram below.



### 9.4.2 Class Relationship/Interaction Diagrams

The following diagram shows the relationship between the individual class diagrams defined in the previous section.



## 9.5 Events

The project consists of events dependent only on the student user.
Student Events:
1) Students log in with NYU credentials
2) Students can view what items they have checkout
3) Students can view the due times of items they have checked out
4) Students can view how many laptops are available
5) Students can view if they have a fine
6) Students can pay their fine

### 9.5.1 Motives

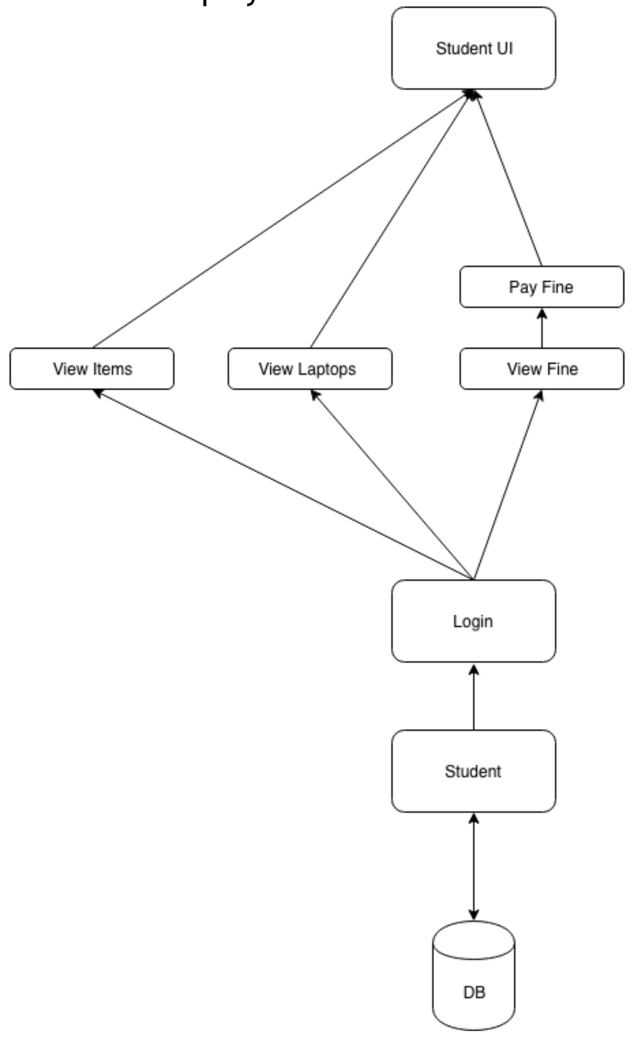
The table below lists the motive for each event.

| Event | Motive |
|---|---|
| Log In | NYU students can log in with their NYU credentials. This account holds their information tied to the Library database. |
| View Items | Students can view their items and their due times so they know when to return them. This is more convenient than viewing it at the front desk of the library or finding it online. |

| | |
|---|---|
| View available laptops | Students can view laptops available so they know whether they can check one out. This is more convenient than viewing it at the front desk of the library. |
| View Fine | Students can view their fine so they know if they have one and how much is due. This is more convenient than viewing it at the front desk of the library. |
| Pay Fine | Students can pay their fine through third party applications. This makes it more convenient that than paying for it over phone. |

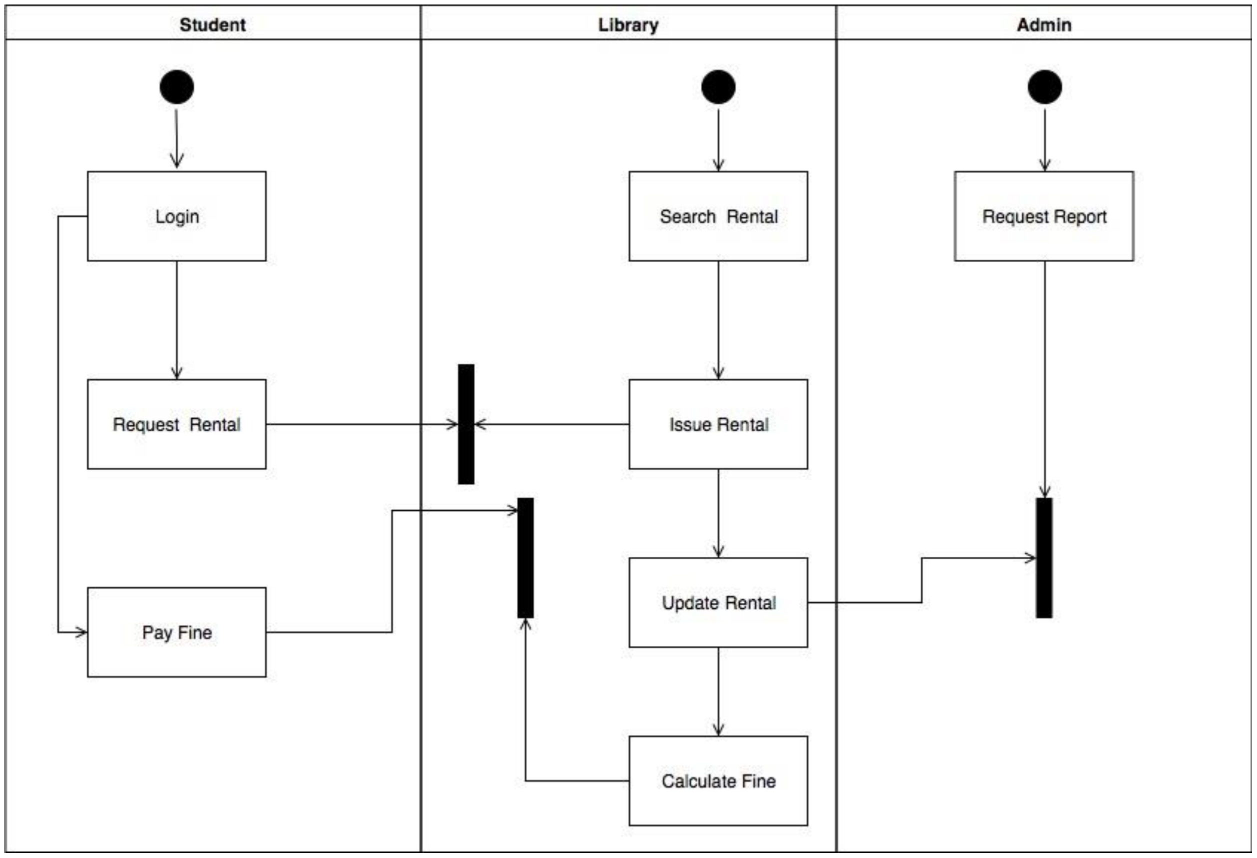
### 9.5.2 Event Diagrams

The flow of Library Tracker is described with the event diagram below. It begins from the student user and every action possible is connected to the student and the database. When the student logs in the information from the library database is filtered through to the student through each action chosen and passed to the UI to be displayed to the student.

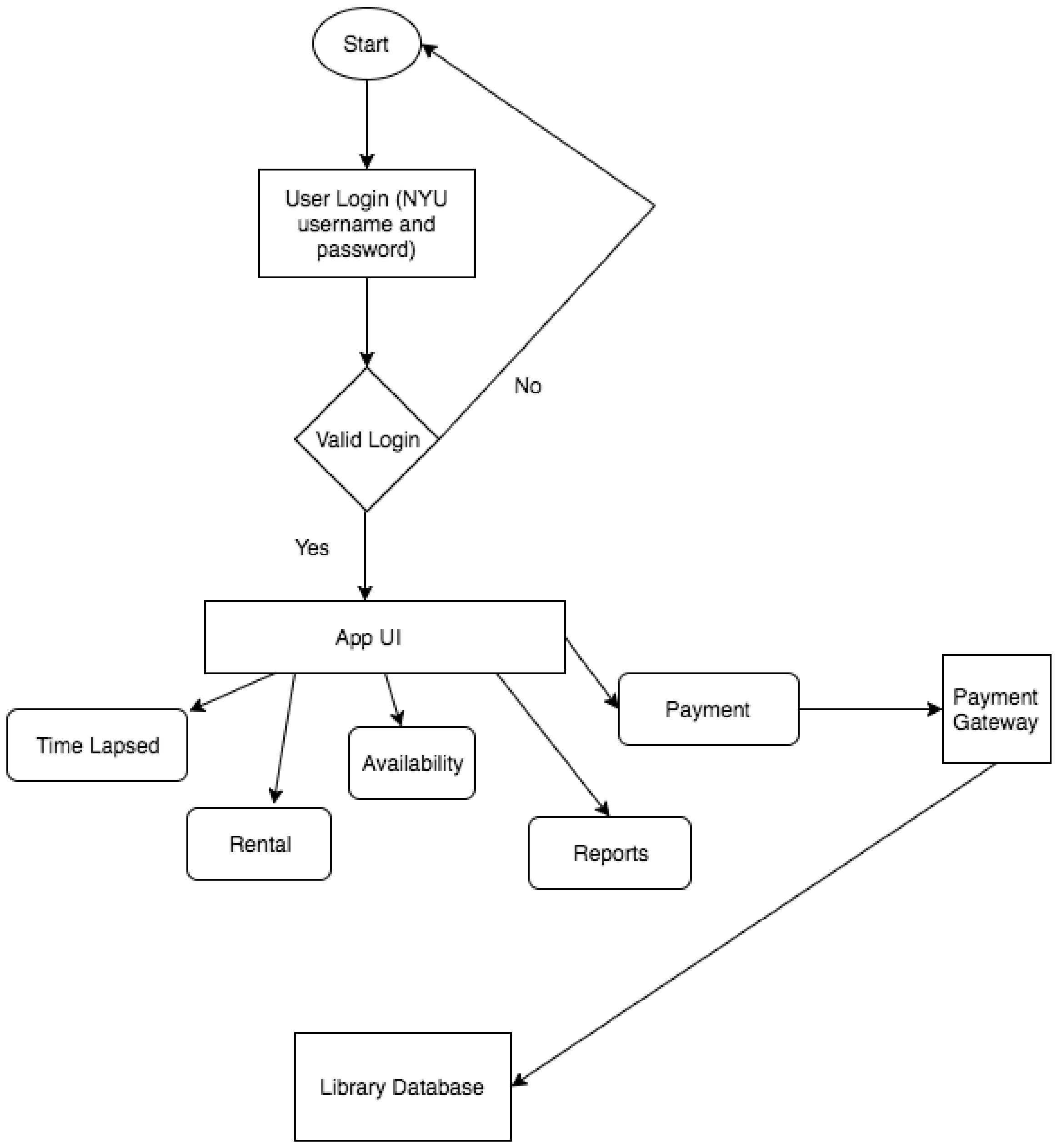


## 9.6 Activity/State (Scenario) Section

The following workflow implements the scenario diagram of the app. This section is further detailed in the Design document.



## 9.7 State Logic



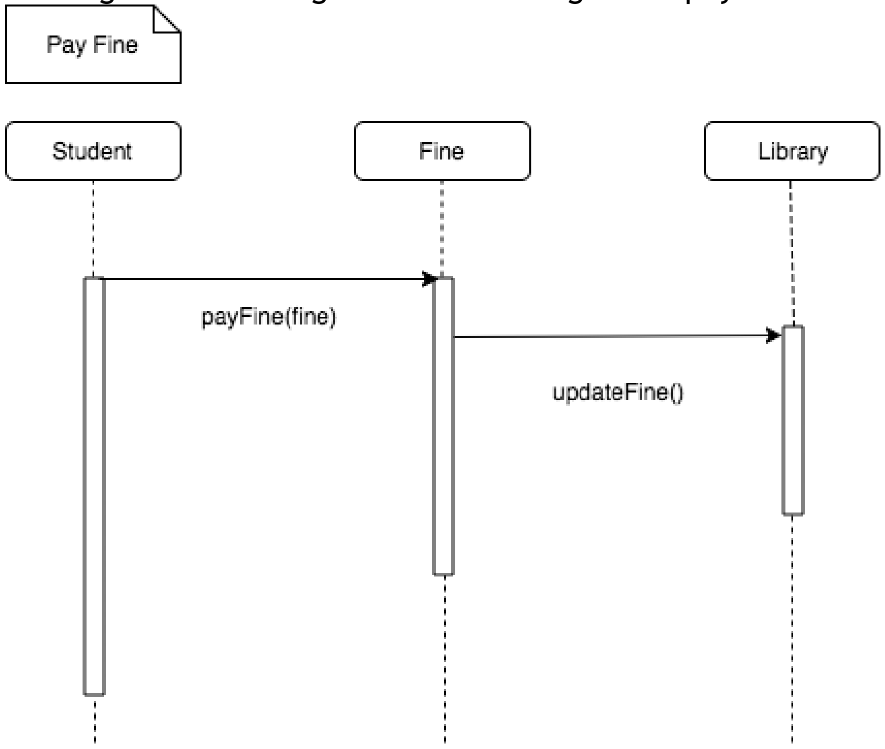
The state logic diagram above models the states within Library Tracker. The user is prompted for their login information including their NYU email and password. If the login information is incorrect, the user will be prompted again to input their login information. If the login information is correct, the user may now access the application's user interface.

## 9.8 Behavior

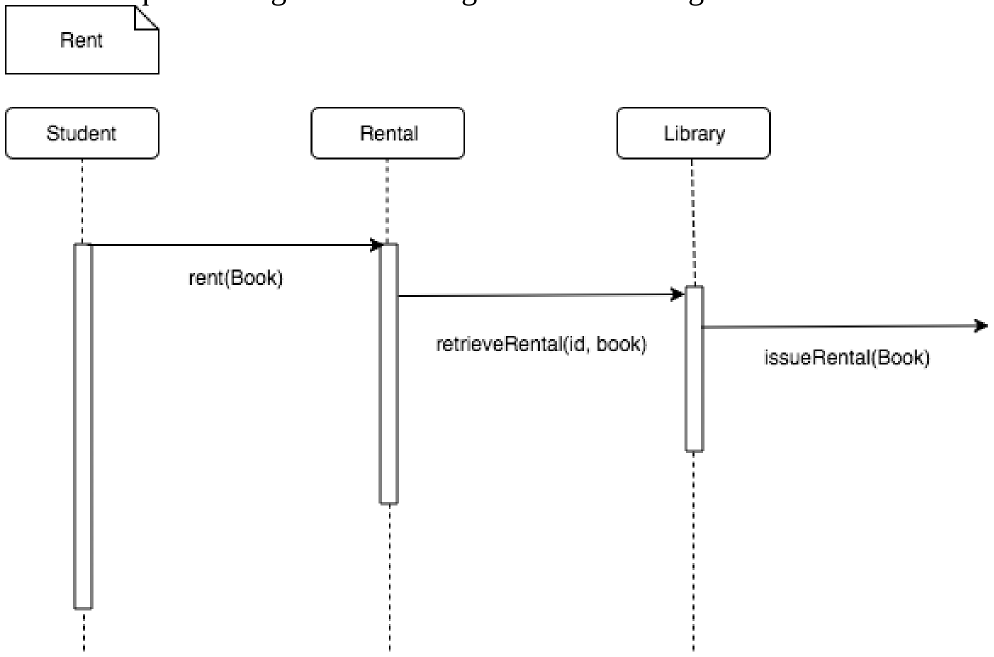
### 9.8.1 Sequence Diagrams

Below are some of the sequence diagrams for this applications. More diagrams will be provided in future documents.

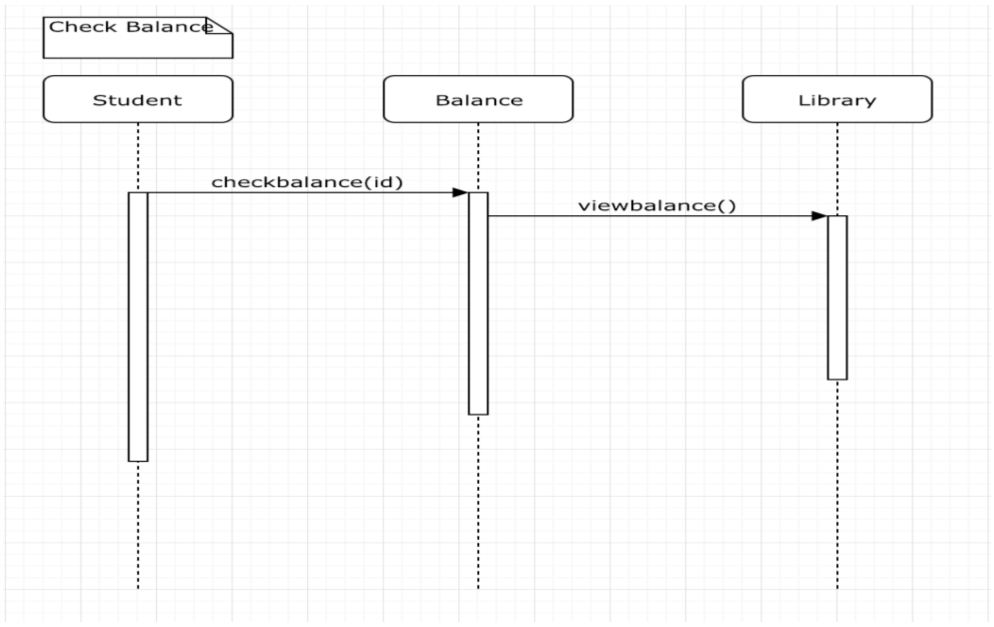
Sequence diagram describing a student making a fine payment.



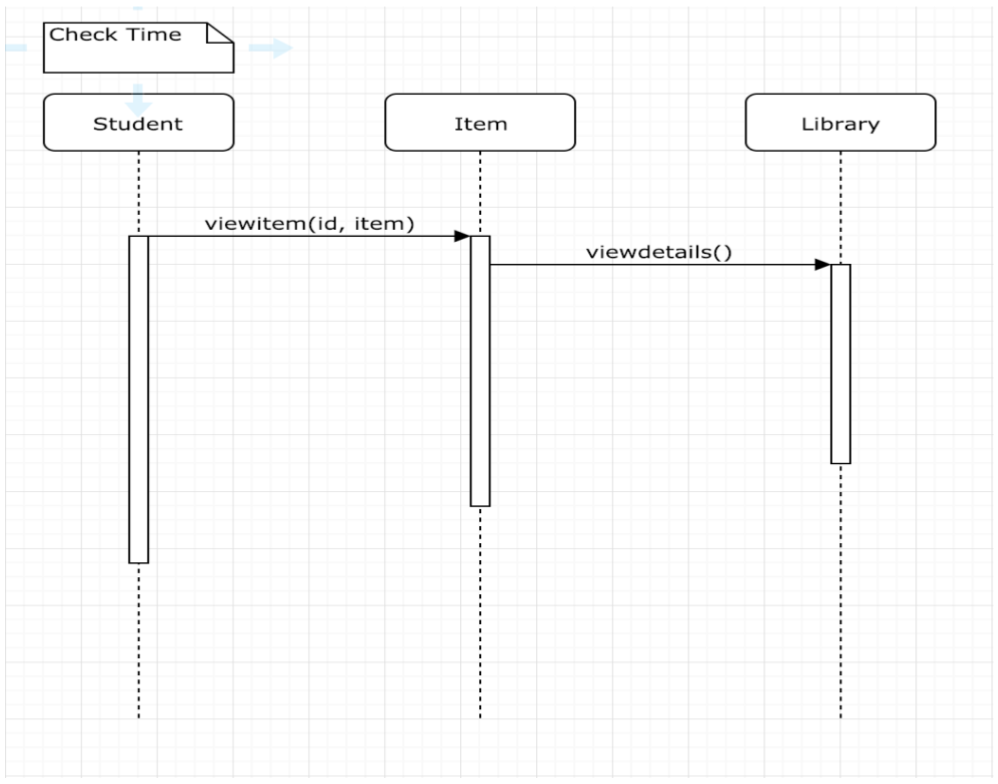
Sequence diagram describing a student making a rental.



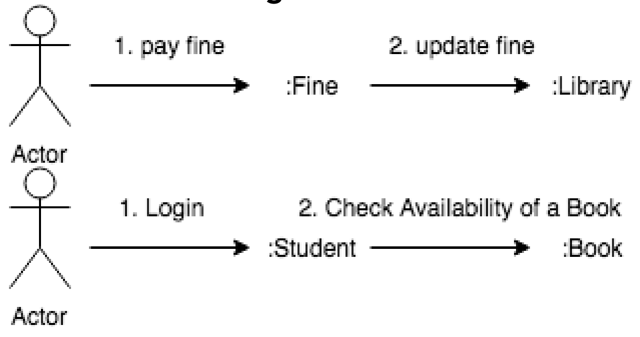
Sequence diagram describing a student checking their balance.



Sequence diagram describing a student checking the time left on an item.



### 9.8.2 Collaboration Diagrams



## 10. SYSTEM TEST PLAN REQUIREMENTS

Test Plan Identifier: Master Test Plan for Library Tracker V1.1

The objective of this test plan will be to ensure that every functional requirement works as planned. The resources that will be used will include xcode or visual studio. The budget constraint for the testing is to keep spending for testing at zero spending. The resources being used maintain for no money to be spent. Limitations of the test plan will be using the actual database of the library as permission is still pending. A separate database will be created to simulate the libraries database. A desk check, a self check, a peer review, a team review, a walkthrough and an inspection will be conducted. SQA will create the test plan, create test scenarios, execute those test scenarios and report defects. The SQA group will moderate the walkthroughs and the inspections. The processes involved in SQA will include a configuration management plan, verification and validation plan, documentation plan, quality assurance plan, reviews and audits, and a problem resolution plan. All members of the group are to participate in the entire SQA process.

## 11. QUALIFICATION PROVISIONS

This document will be reviewed for quality by the team and peer review. The review will be to check if there are any missing information or incomplete sections. Any unclear sections will also be modified if needed. The four types of testing that will be done will include unit testing, integration testing, system testing and interface testing. The process by which this document will be verified for quality will be through checking that the document meets certain requirements. The requirements include if the document is correct, unambiguous, complete, consistent, stable, verifiable, modifiable, traceable. The verification and validation plan involves desk checks, peer reviews, and inspections. Peer reviews and desk checks will occur every cycle between members to inspect for errors and trace the logic in source code and documentations. Inspections will occur every other week to ensure the project schedule is met and allow prioritization of tasks. Formal technical reviews will occur after major milestones are met. Further plans for inspection and quality insurance will come from an external SQA team.

## 12. REQUIREMENTS TRACEABILITY

Each requirement will be thoroughly documented for the entire project and will have a unique identifier that will be consistent throughout the documentation and code. Therefore, when a defect is found, its path can be followed all the way up to the insertion of that specific requirement. In doing so, it will be easier to fully identify the defect and to know what changes to make in order to fix said defect. Two types of traceability are forward traceability which leads to all artifacts spawned by this document and backward traceability which leads to previous stages of development in the project.

## 13. EVOLUTION OF THE RAS

The initial release will be distributed and any changes made will be documented. The document itself will be updated any time a change occurs and will be inspected each time a new version is produced. Each time a new version is produced and inspected, the document will be released with a new version number. Any additional changes may result from deficiencies, shortcomings, inaccuracies, or changes in the system environment. Essentially, this document evolves as the system development process progresses. In order to identify control, track, and report projected changes, a formal change process will be initiated. Approved changes, as a result, will be incorporate in this document to ensure that there is an accurate and complete trail of changes and to allow there to be a review of current and superseded portions of the document.

## 14. RATIONALE

## 15. NOTES

## 16. APPENDICES

## SCHEDULE TRACKING

| Artifact or Deliverable | Who | Estimated | Actual | Difference |
|---|---|---|---|---|
| RAS 1.0 | Girish | 3 | 2 | 1 |
| RAS 1.0 | Ricardo | 1 | 2 | 1 |
| RAS 1.0 | Viola | 3 | 1 | 2 |
|  | Summary for entire team | 7 | 6 | 4 |

| Artifact or Deliverable | Who | Estimated | Actual | Difference |
|---|---|---|---|---|
| RAS 1.1 | Girish | 3 | 2 | 1 |
| RAS 1.1 | Ricardo | 1 | 2 | 1 |
| RAS 1.1 | Viola | 3 | 1 | 2 |
|  | Summary for entire team | 7 | 6 | 4 |

| Artifact or Deliverable | Who | Estimated | Actual | Difference |
|---|---|---|---|---|
| SPMP | Girish | 2 | 2 | 0 |
| SPMP | Ricardo | 1 | 1 | 0 |
| SPMP | Viola | 2 | 1 | 1 |
|  | Summary for entire team | 5 | 4 | 1 |

**Cumulative**

| Artifact or Deliverable | Who | Estimated | Actual | Difference |
|---|---|---|---|---|
| Project Proposal | Team B27 | 2 | 2 | 0 |
| RAS 1.0 | Team B27 | 7 | 6 | 4 |
| RAS 1.1 | Team B27 | 7 | 6 | 4 |
| SPMP | Team B27 | 5 | 4 | 1 |

## DEFECT TRACKING

| Artifact or Deliverable | Who | Estimated | Actual | Difference |
|---|---|---|---|---|
| RAS 1.0 | Girish | 3 | 2 | 1 |
| RAS 1.0 | Ricardo | 1 | 2 | 1 |
| RAS 1.0 | Viola | 3 | 1 | 2 |
|  | Summary for entire team | 7 | 6 | 4 |

| Artifact or Deliverable | Who | Estimated | Actual | Difference |
|---|---|---|---|---|
| RAS 1.1 | Girish | 3 | 2 | 1 |
| RAS 1.1 | Ricardo | 1 | 2 | 1 |
| RAS 1.1 | Viola | 3 | 1 | 2 |
|  | Summary for entire team | 7 | 6 | 4 |

| Artifact or Deliverable | Who | Estimated | Actual | Difference |
|---|---|---|---|---|
| SPMP | Girish | 2 | 2 | 0 |
| SPMP | Ricardo | 1 | 1 | 0 |
| SPMP | Viola | 2 | 1 | 1 |
| | Summary for entire team | 5 | 4 | 1 |

**Cumulative**

| Artifact or Deliverable | Who | Estimated | Actual | Difference |
|---|---|---|---|---|
| Project Proposal | Team B27 | 2 | 2 | 0 |
| RAS 1.0 | Team B27 | 7 | 6 | 4 |
| RAS 1.1 | Team B27 | 7 | 6 | 4 |
| SPMP | Team B27 | 5 | 4 | 1 |