System Design Document

To be submitted to the Department of Mathematics and Computer Science, Gordon College
in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science

by:
Eze Anyanwu,
Stephanie Powers,
Weiqiu (Rachel) You

Document accepted on _____ by _____
(date) (client)

Document accepted on _____ by _____
(date) (departmental representative)

1. Introduction

1.1 Purpose of the system

The Electronic Room Condition Inventory system will benefit students, Resident Advisors, and Resident Directors by providing a web-based system with a well thought-out user interface that eliminates the redundancy, ambiguity, tediousness, and wastefulness of the current paper RCI system.

1.2 Scope of the system

The key functionality of the system will be a user interface for students to input information about their rooms, and an interface for the RA to input charges corresponding to the room condition at the end of the year. There will also be an RD view, which would allow the RD to determine who has not completed their RCI's, as well as to export a spreadsheet with residents' fines at the end of the year. Any user designated as an admin will have access to and control over all students' RCI's as well.

1.3 Objectives and success criteria of the project

Our objective is to create an electronic system for the process of collecting and processing room condition data from residents in the residence halls of Gordon College. We will be successful if, at the end of the development of the project, we have created a deliverable to the Housing Department that will include a web application that allows residents to input their RCI data electronically, and that allows RA's to input fines at the end of the year. To be successful, the system must also allow some level of interfacing for the RD to be able to view all this info collected and export a fines spreadsheet.

1.4 Definitions, acronyms, and abbreviations

- **RCI**: Room Condition Inventory
- **RA**: Resident Advisor
- **RD**: Resident Director

2. Current system

The current system for documenting the condition of rooms is redundant, ambiguous and tedious. More specifically, students use an RCI (Room Condition Inventory) to document the state of the room when they move in and move out. The RCI is a long double-sided sheet of paper with space to comment on the state of various pieces of furniture in the room.

As a paper-based system, it has become hard to manage for the hundreds of students who live on campus. It is ambiguous because the descriptions are subjective. What one student sees as major damage might be seen by another as minor. Accurately describing the extent of the damage also gets tricky. It is common during move-out procedures for students to protest fines for damages by saying "It was there before!".

It is tedious because a lot of manual work has to be put in by Residence Life to make sure information gets gathered.

3. Proposed system

3.1 Overview

The Online RCI system will provide a web user interface for students, RA's, and RD's to record RCI data.

3.2 Functional requirements

The Online RCI system will provide a web user interface for students to input information about their rooms, using a combination of free forms and photo uploads. Residents of an apartment, which includes a common area, will also have a common area RCI that will be editable by all members of the apartment. The system will also include an interface for the RA to input charges corresponding to the room condition at the end of the year, and for the RD to view and sort all the data coming in from their resident hall. RA's and RD's will have access to all the RCI's for all the rooms of their building, and will be able to designate which components within a room belong to which residents.

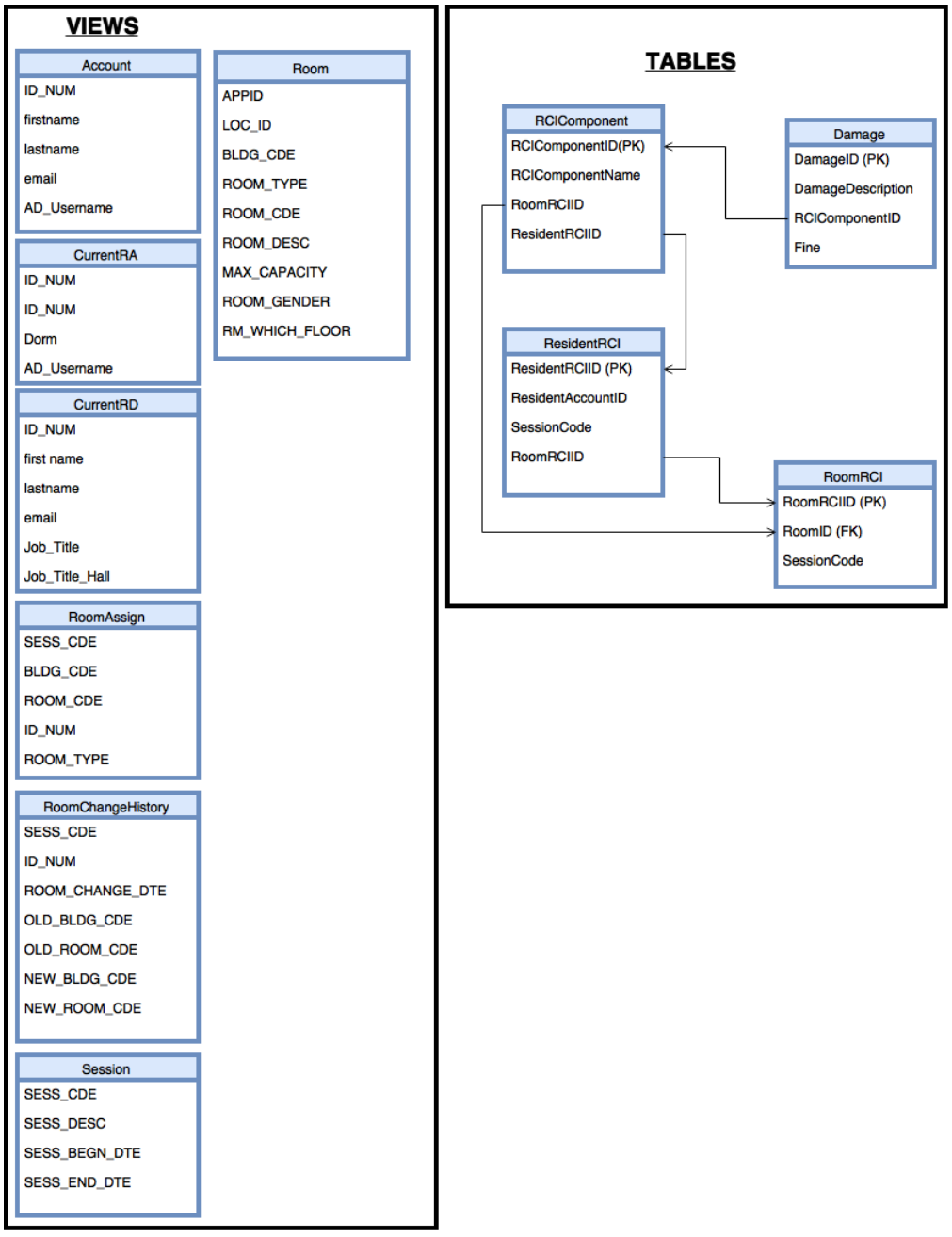
3.3 Non-functional requirements

- *3.3.1 Usability*: User should be able to navigate the site without much coaching. A basic user guide will be provided. User interface will be responsive for easy mobile use.
- *3.3.2 Reliability*: Site will function over HTTPS.
- *3.3.3 Performance*: Response Time will be typical for a web app.
- *3.3.4 Supportability*: System designed such that NSG can maintain it.
- *3.3.5 Implementation*: Will be determined in conjunction with CTS.
- *3.3.6 Interface*: Format of Fines Document must be acceptable by Student Financial Services.
- *3.3.7 Packaging*: Web application will be served from the Gordon CS Department VM's.

3.3.8 Legal: The signing process must be approved by the office of Student Life.

4. Design Diagrams:

4.1.1 Database Schema



4.1.2 Database Schema Notes

Since we will hopefully be handing this off to CTS in the future, we are trying to use the technologies they are most familiar with. So for persistence, we will be using a Microsoft SQL Server.

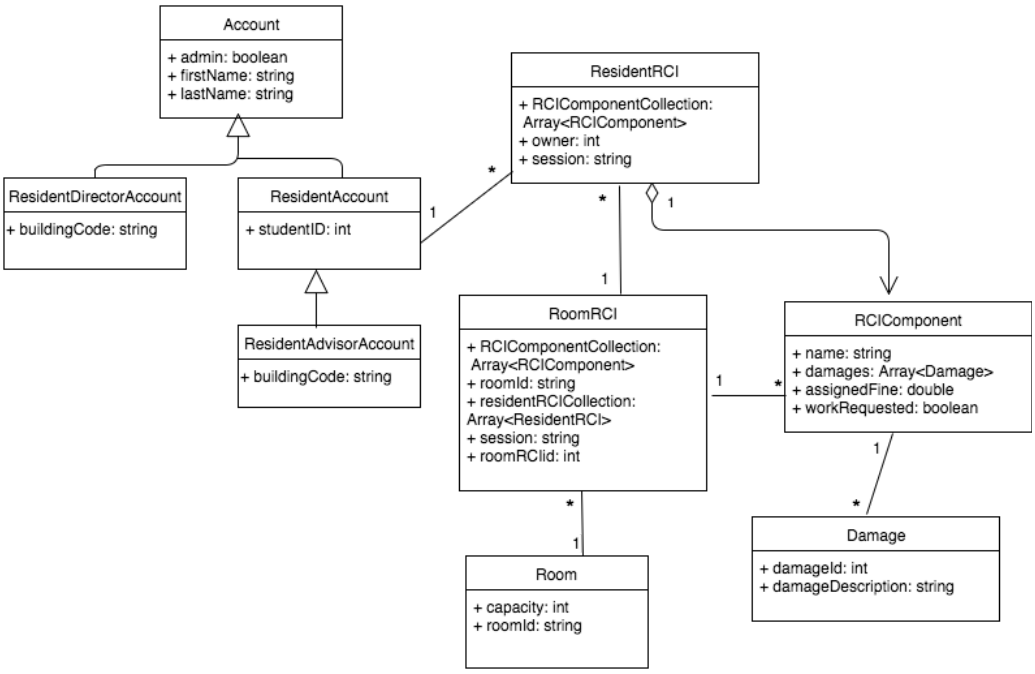
Views: The contents of the views should be obvious by their names. These are views into various parts of the Gordon database and were provided by CTS. SQL doesn't allow using a column in a View as the foreign key for a table. This is why the views stand apart with no relationships. Thankfully, we can write insert and update triggers for the tables that check referential integrity.

ORM: The creation of this table will be done through an Object Relational Mapper called Entity Framework. This means that the C# classes we create will closely correspond to the tables we have.

Tables:

- *RoomRCI*: Represents the RCI for an entire room regardless of how many residents are in it. RCIComponent has a foreign key to this table to indicate that the component is associated to this room.
- *ResidentRCI*: Represents the RCI for a single student. RCIComponent has a foreign key to this table to indicate that the component is associated with this student.
- *RCIComponent*: One of the two important tables. Each record is a piece of furniture whose state can be documented (e.g. table, desk, wall, carpet …). Damage has a foreign key to this table to indicate that the damage is
- *Damage*: The second important table. Records damage descriptions that users enter. Links each description to its component.

4.2.1 Class Diagram

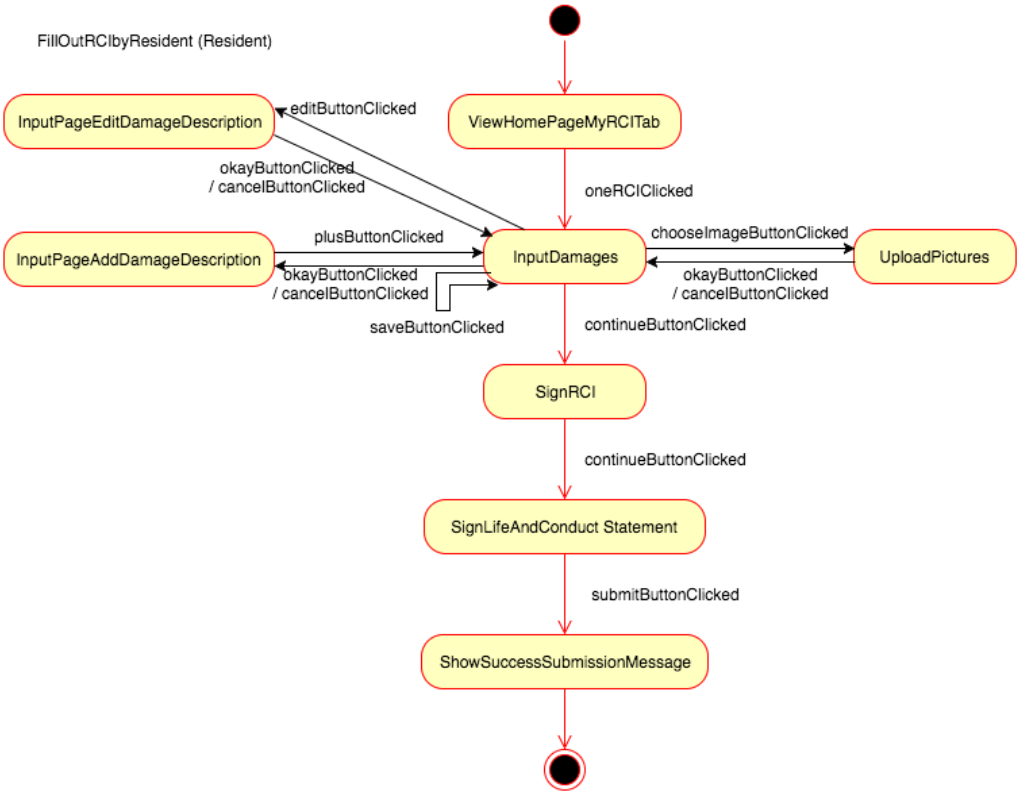


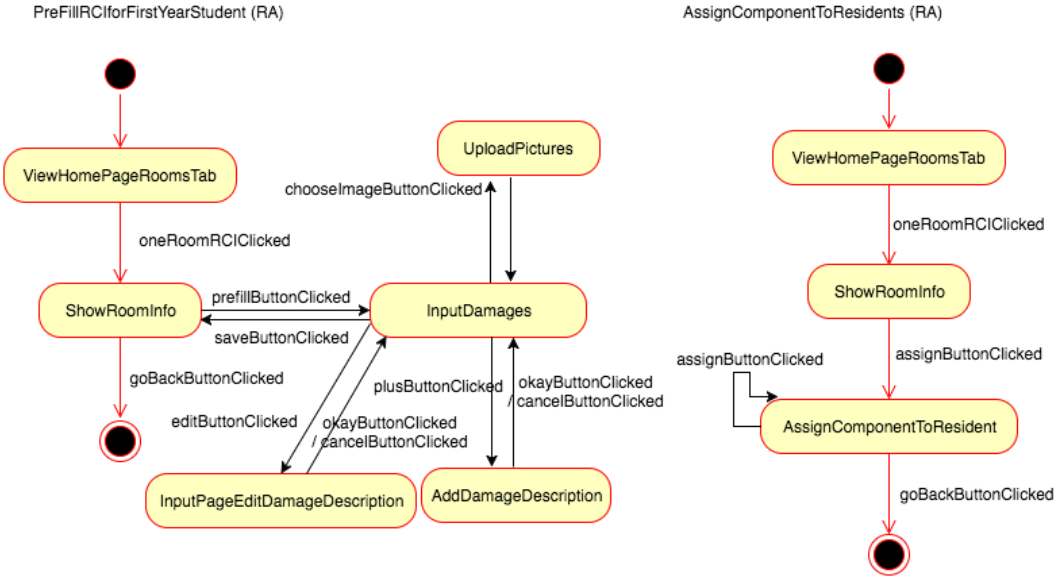
4.2.2 Class Descriptions:

- `Account`: represents a user as an account object, with firstName and lastName string attributes, as well as the boolean admin to indicate if an user should be given admin privileges or not
- `ResidentDirectorAccount` : inherits from `Account` and has a buildingCode attribute to denote what building the RD should have access to in the system
- `ResidentAccount` : inherits from `Account` and has a studentId attribute
- `ResidentAdvisorAccount`: Inherits from `ResidentAccount`, given that RA's are also residents themselves. Like the RD, RA's will have a buildingCode attribute denoting what building they should have access to
- `ResidentRCI`: represents a single resident's collection of room components during a single year in a single dorm. This object strongly relates to the current RCI paper form that residents fill out. Every `ResidentAccount` could have potentially many `ResidentRCI` objects associated with it, from year to year or even within a semester.
- `RCIComponent` : represents each element within a room (e.g. desk, carpet) and its corresponding collection of damages and fines, as well as the boolean workRequested to represent whether or not a workRequest has been filed for a given component.

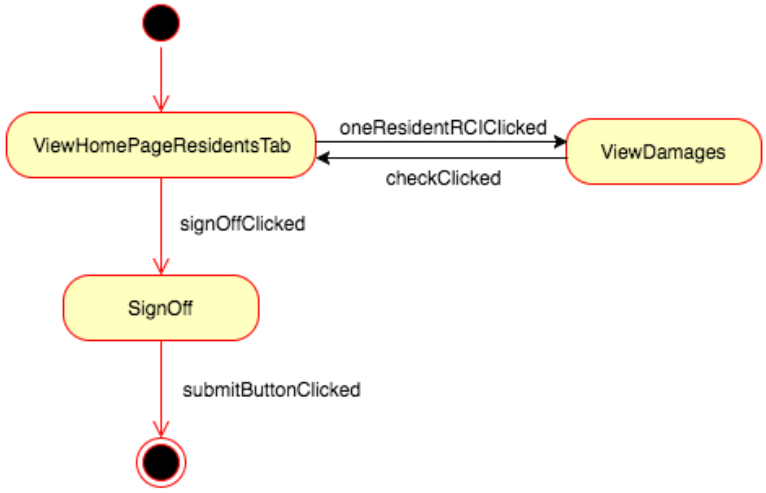
- `RoomRCI`: allows room components to belong just to the room, particularly relevant in the case of an RA pre-filling RCI's for freshmen who have not yet moved in, to claim which room components belong to whom.
- `Room`: represents the physical room, which stays relatively constant. Enables us to keep a history of all the `RoomRCI` objects associated with a given room.
- `Damage`: represents an individual damage for an individual component. Particularly useful in the case in which a component may have multiple damages.

4.3.1 State Diagram

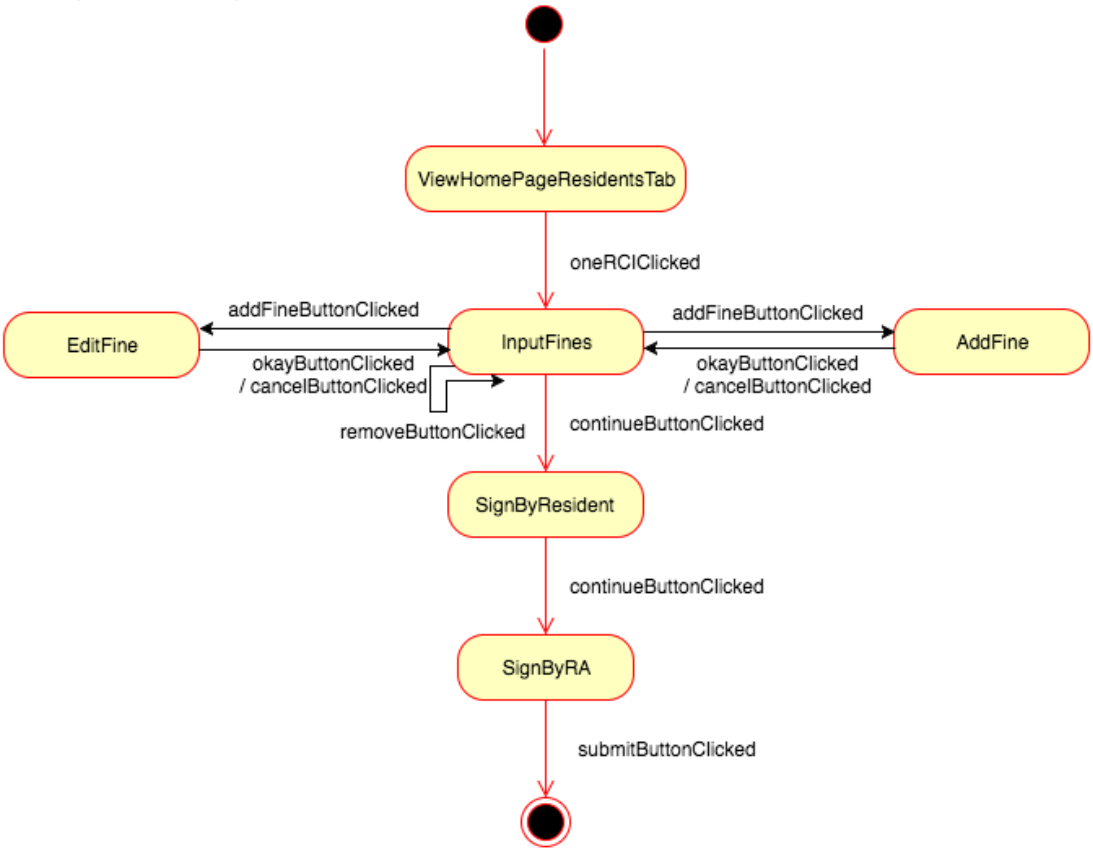


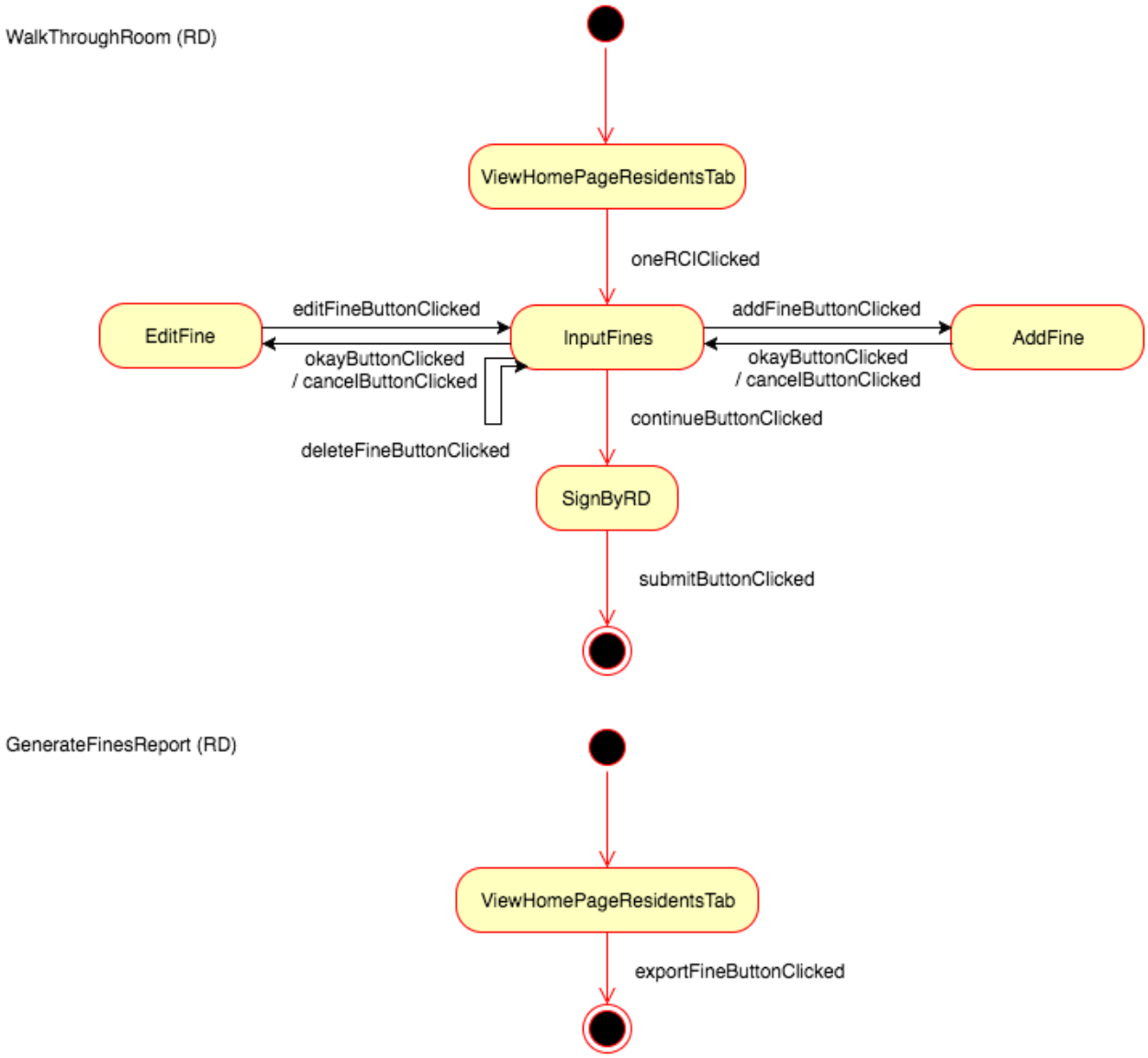


ReviewRCIbyRD (RD)



CheckOutResident (RA, with Resident)





4.3.2 State Diagram Notes

The seven state diagrams below each represents one use case.

In *FillOutRCIbyResident*, a resident opens an individual RCI and start inputting damages. The resident can add, edit, and delete damages for components for his/her side of the room. The resident can save progress at any time. After signing the RCI and Life and Conduct Statement, the resident has completed one RCI.

In *PreFillRCIforFirstYearStudent*, an RA opens a room RCI and start inputting damages. The RA can add, edit, and delete damages for components of the whole room. The RA can save progress at any time.

In *AssignComponentToResidents*, after prefilling damages for components in a room an RA opens the room RCI and select one item of each category to assign to one resident at a time. The RA can do so multiple times until all components are assigned to residents in the room.

In *ReviewRCIbyRD*, an RD opens an individual RCI at a time and check “accepted” if the RCI is acceptable. Then the RD can go to sign off view from the home page to sign off all checked RCI’s at one time.

In *CheckOutResident*, an RA opens an individual RCI and starting inputting fines. The RA can add, edit, and delete fines for components in the RCI that have new damages. The RA can save progress at any time. After the resident signing and then the RA signing, the checkout process is complete.

In *WalkThroughRoom*, an RD opens an individual RCI and starting checking fines that an RA has inputted. The RD can add, edit, and delete fines for components in the RCI. The RD can save progress at any time. After the RD signing, the walk through process is completed.

In *GenerateFinesReport*, an RD selects some of or all of the RCI’s on the home page and clicks “Generate Fines” button, and fines of the selected RCI’s are exported in an EXCEL file.