Project: Online RCI
Group Members: Stephanie Powers, Weiqiu You, Eze Anyanwu

Initial System Structure

The RCI system will make use of the MVC design pattern:

Model

We represent the different actors and entities in the system using classes. We use inheritance to represent the abstraction inherent in the current RCI system.

- Class Account:
  - Class ResidentAccount extends Account
  - Class ResidentAdvisorAccount extends Account
  - Class ResidentDirectorAccount extends Account
- Class RCI:
  - Class HUDRCI extends RCI
  - Class FerrinDrewRCI extends RCI
  - Class CommonAreaRCI extends RCI
- Class Room: (example attributes: room-number, capacity, inhabitants, building)

View

Directly correlates with our User Interface mockup. Will be generated using Razor, a markup engine associated with ASP.NET

Controller

The correlation between controllers and views will be almost 1:1. For most cases, there will be a unique controller behind each view. With this in mind, we can outline the following potential controller classes:

- Class AuthenticationController - Handles authentication
- Class DashboardController - Handles any operations that can be done through the dashboard
- Class FurnitureAssignController - Handles assigning pieces of furniture to specific students.
- Class RCIInputController - Handles completion of an RCI
- Class RCICheckoutController - Handles end-of-year checkout
- Class RCIWalkthroughController - Handles end-of-year walkthrough

Subsystems

The RCI system can be logically decomposed into the following subsystems:

- *Authentication Subsystem*
    - Handles user login and logout. Will interface with CTS's preexisting LDAP authentication system, with HTTPS to ensure security logging in for users.
    - Potential modularity breakdown:
        - login
        - logout

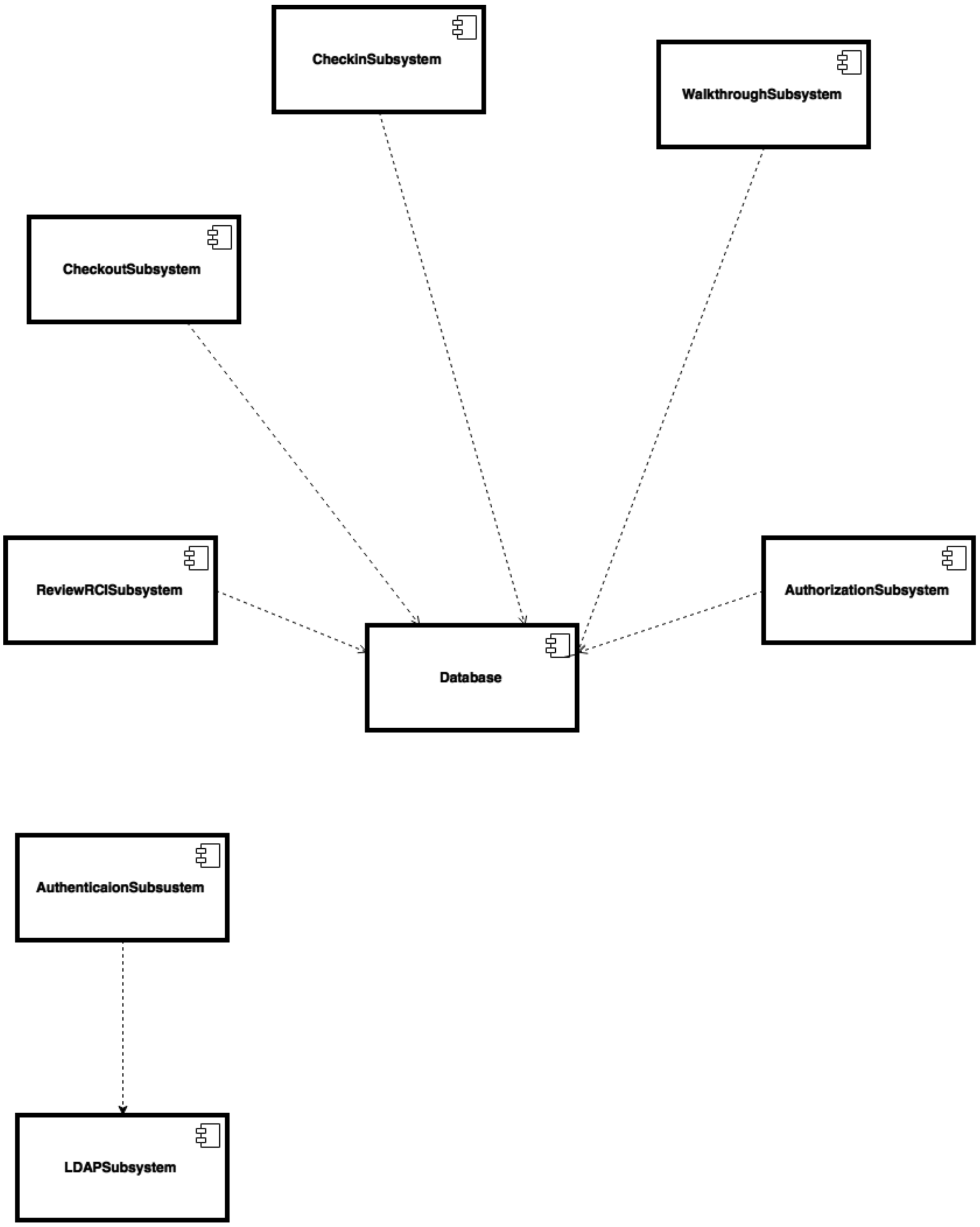
- *Authorization Subsystem*
    - Granting user access control to certain RCI's and certain parts of RCI's (who-has-which permissions)
    - Potential modularity breakdown for each type of user's access:
        - Resident: self RCI, (possible) common area RCI; entering damages
        - RA: self RCI, (possible) common area RCI, RCI's of residents in the building, RCI's of rooms in the building; entering fines
        - RD: RCI's of residents in the building; entering fines

- *Checkin Subsystem*
    - Handles the process of a resident checking in, as described in the `PreFillRCIforFirstYearStudent` and `FillOutRCIbyResident` use case, coordinating various modules of functionality, seen below:
    - Potential modularity breakdown:
        - RA prefill RCI for first year students by filling out damages for the whole room, and assign furnitures to students by querying them about which resident has chosen which furnitures
        - Resident entering damages
        - RA signoff
    - Review RCI Sub-subsystem (Extended from Checkin Subsystem)
        - RD signs off a bunch of RCI's at once after they are signed off by RA.

- *Checkout Subsystem*
    - Handles the process of a resident checking out, as described in the `CheckoutResident` use case, coordinating various modules of functionality, seen below
    - Potential modularity breakdown:
        - Entering fines
        - Resident signoff
        - RA signoff

- *Walkthrough Subsystem*
  - Handles the process of an RA and RD walking through a room after Resident has checked out, as described in the `WalkThroughRoom` use case, coordinating various modules of modularity, seen below
  - Potential modularity breakdown
    - Loading all the rooms for which checkouts have been completed
    - Editing/adding fines
    - RD signoff
- *Database*
  - Stores all user data; will be a relational database, already provided by CTS
  - Current views include:
    - ACCOUNT - View of user login information needed for LDAP authentication within your application. (Eze did this with a previous transcript project and will know what is needed here.)
    - CM_SESSION_MASTER - View of all available Sessions with their sess_cde, sess_desc, sess_begin_dte, & sess_end_dte
    - ROOM_MASTER – View of all rooms with their LOC_CDE, BLDG_CDE, and all other applicable fields related to a room.
    - ROOM_ASSIGN – View of all room assignments based on SESS_CDE, LOC_CDE, and BLDG_CDE. The ID_NUM field is the Gordon ID of the person assigned to the room.
    - ROOM_CHANGE_HIST – View of room change history in the middle of a session.
    - CURRENT_RDS - View of all the current RDs and their building assignments.

Initial Subsystem Decomposition


CheckinSubsystem
WalkthroughSubsystem
CheckoutSubsystem
ReviewRCISubsystem
Database
AuthorizationSubsystem
AuthenticaionSubsustem
LDAPSubsystem