Sydney

Bennett, Jefferson, Jenny, Kelly, Matt

Agenda

- Problem & Our Solution
- Tech-Stack
- Documentation (ERD, Class Diagrams)
- Demo
- Testing
- Reflections
- Questions

The Problem

Document computers along with its installed software and requirements for different roles within the academy

Employees want to know which computer is available and appropriate to use according to their role

Our Solution

PC Track

Software inventory and access management system

- Web application for managing the software availability
- Database to store information which would then be retrieved to access from the website
- Secure access system for different user roles

Tech Stack Used

Front-End

- React.js
- CSS
- VS Code

Back-End

- Spring.io
- Eclipse
- Java

Database Tools

- MySQL
- Postman

Tech Stack Used

Package Manager

pnpm

Version Control

Git Bash

GitLab

GitHub Desktop

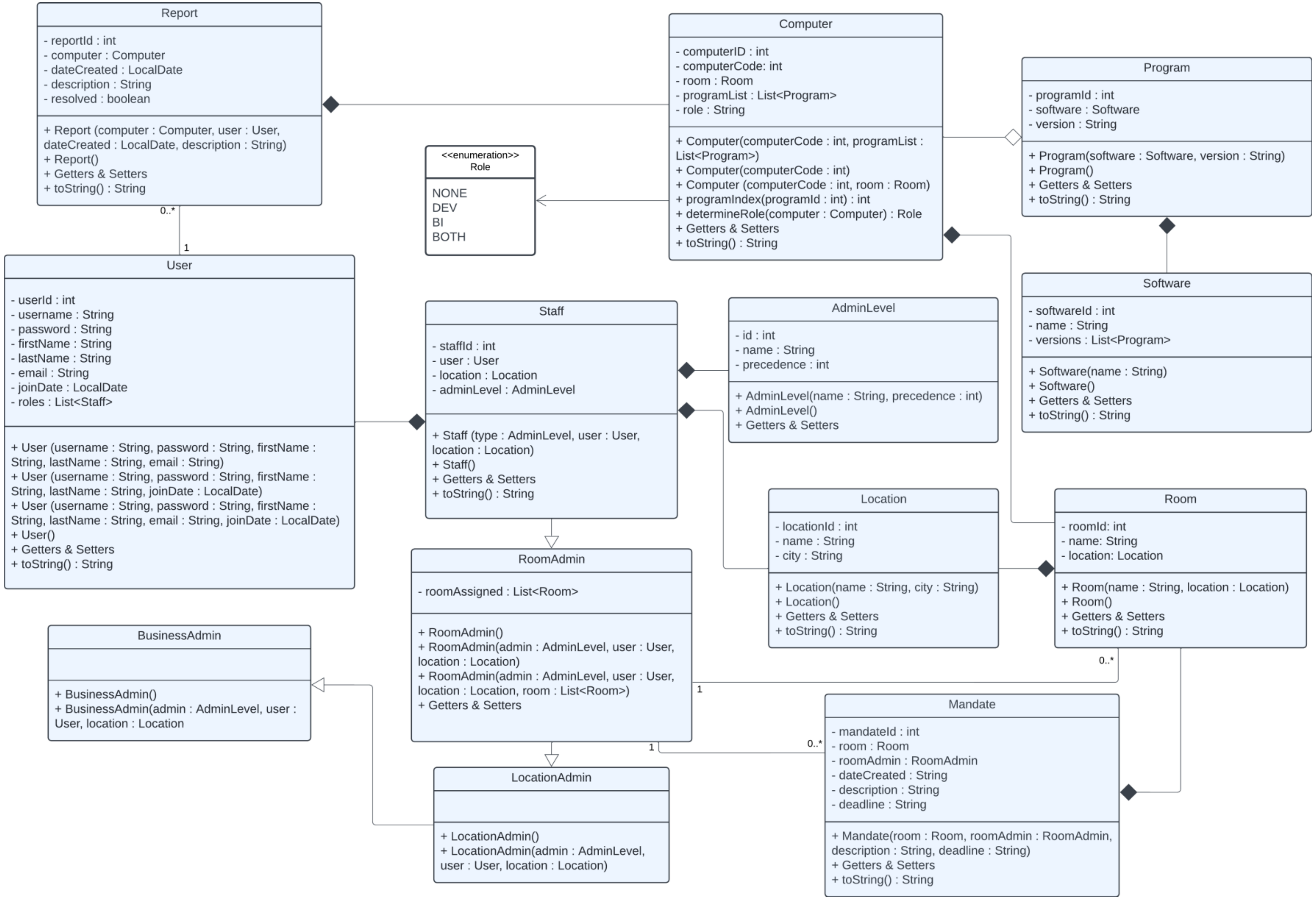
Packages Used

React Router

React-router-dom

Axios

Material UI



Report

- reportId : int
- computer : Computer
- dateCreated : LocalDate
- description : String
- resolved : boolean

+ Report (computer : Computer, user : User, dateCreated : LocalDate, description : String)
+ Report()
+ Getters & Setters
+ toString() : String

Computer

- computerID : int
- computerCode: int
- room : Room
- programList : List<Program>
- role : String

+ Computer(computerCode : int, programList : List<Program>)
+ Computer(computerCode : int)
+ Computer (computerCode : int, room : Room)
+ programIndex(programId : int) : int
+ determineRole(computer : Computer) : Role
+ Getters & Setters
+ toString() : String

Program

- programId : int
- software : Software
- version : String

+ Program(software : Software, version : String)
+ Program()
+ Getters & Setters
+ toString() : String

<<enumeration>>
Role

NONE
DEV
BI
BOTH

User

- userId : int
- username : String
- password : String
- firstName : String
- lastName : String
- email : String
- joinDate : LocalDate
- roles : List<Staff>

+ User (username : String, password : String, firstName : String, lastName : String, email : String)
+ User (username : String, password : String, firstName : String, lastName : String, joinDate : LocalDate)
+ User (username : String, password : String, firstName : String, lastName : String, email : String, joinDate : LocalDate)
+ User()
+ Getters & Setters
+ toString() : String

Staff

- staffId : int
- user : User
- location : Location
- adminLevel : AdminLevel

+ Staff (type : AdminLevel, user : User, location : Location)
+ Staff()
+ Getters & Setters
+ toString() : String

AdminLevel

- id : int
- name : String
- precedence : int

+ AdminLevel(name : String, precedence : int)
+ AdminLevel()
+ Getters & Setters

Software

- softwareId : int
- name : String
- versions : List<Program>

+ Software(name : String)
+ Software()
+ Getters & Setters
+ toString() : String

Location

- locationId : int
- name : String
- city : String

+ Location(name : String, city : String)
+ Location()
+ Getters & Setters
+ toString() : String

Room

- roomId: int
- name: String
- location: Location

+ Room(name : String, location : Location)
+ Room()
+ Getters & Setters
+ toString() : String

RoomAdmin

- roomAssigned : List<Room>

+ RoomAdmin()
+ RoomAdmin(admin : AdminLevel, user : User, location : Location)
+ RoomAdmin(admin : AdminLevel, user : User, location : Location, room : List<Room>)
+ Getters & Setters

BusinessAdmin

+ BusinessAdmin()
+ BusinessAdmin(admin : AdminLevel, user : User, location : Location

Mandate

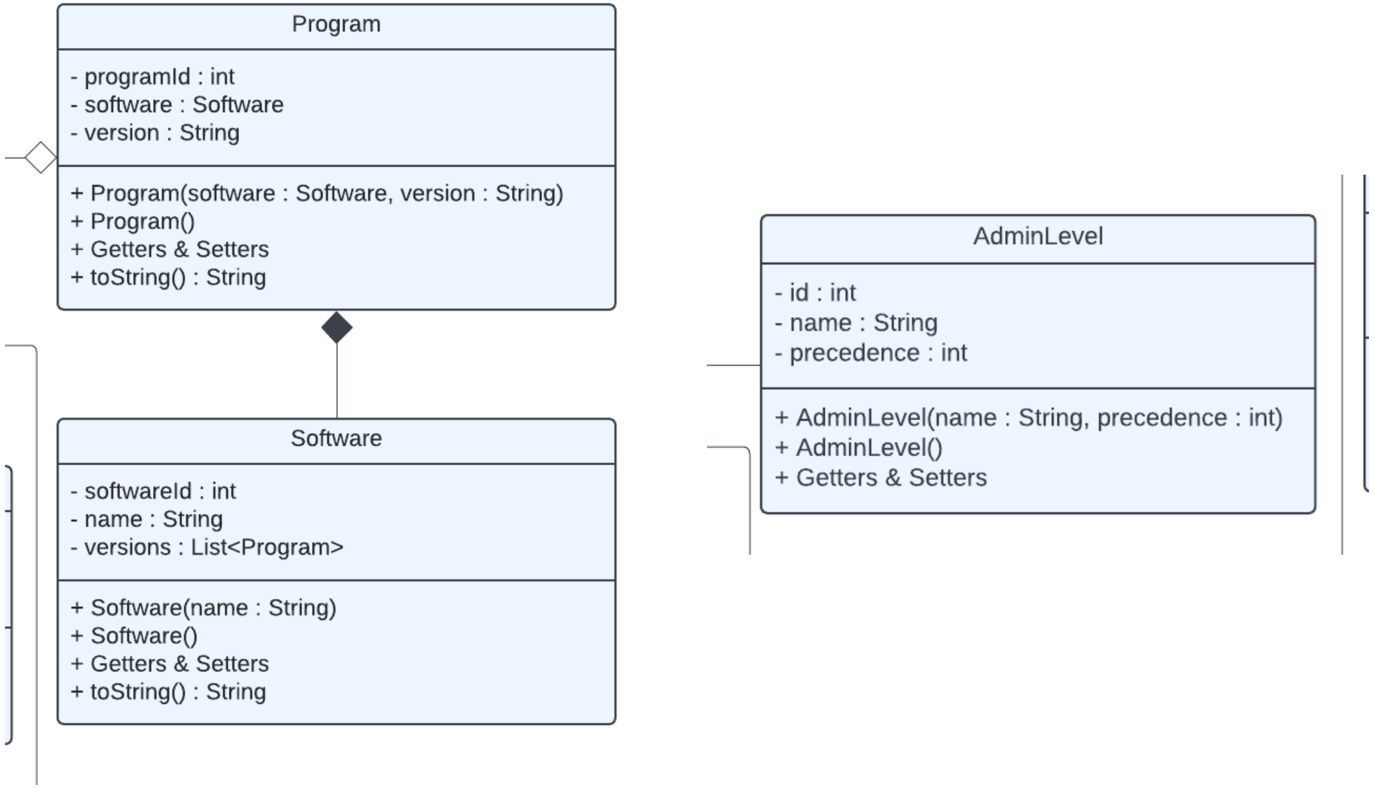
- mandateId : int
- room : Room
- roomAdmin : RoomAdmin
- dateCreated : String
- description : String
- deadline : String

+ Mandate(room : Room, roomAdmin : RoomAdmin, description : String, deadline : String)
+ Getters & Setters
+ toString() : String

LocationAdmin

+ LocationAdmin()
+ LocationAdmin(admin : AdminLevel, user : User, location : Location)

- Staff now includes an AdminLevel class
  - Display admin role and precedence in displayable datatype
  - Mainly used for front-end query
- Program is now split to program and software class
  - Software holds the name
  - Program holds the version number and its corresponding software



Program

- programId : int
- software : Software
- version : String

+ Program(software : Software, version : String)
+ Program()
+ Getters & Setters
+ toString() : String

Software

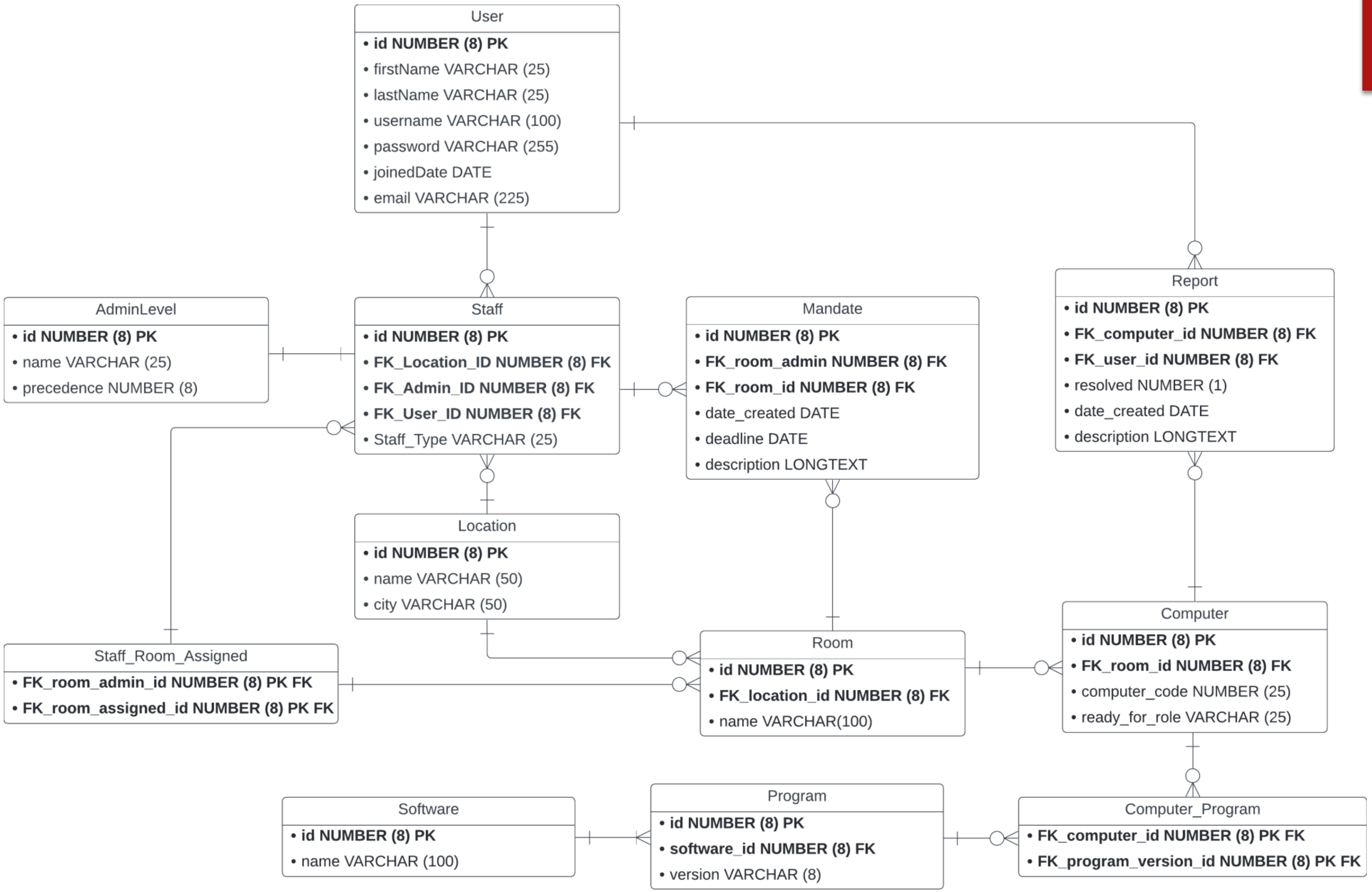
- softwareId : int
- name : String
- versions : List<Program>

+ Software(name : String)
+ Software()
+ Getters & Setters
+ toString() : String

AdminLevel

- id : int
- name : String
- precedence : int

+ AdminLevel(name : String, precedence : int)
+ AdminLevel()
+ Getters & Setters



User
- **id NUMBER (8) PK**
- firstName VARCHAR (25)
- lastName VARCHAR (25)
- username VARCHAR (100)
- password VARCHAR (255)
- joinedDate DATE
- email VARCHAR (225)

AdminLevel
- **id NUMBER (8) PK**
- name VARCHAR (25)
- precedence NUMBER (8)

Staff
- **id NUMBER (8) PK**
- **FK_Location_ID NUMBER (8) FK**
- **FK_Admin_ID NUMBER (8) FK**
- **FK_User_ID NUMBER (8) FK**
- Staff_Type VARCHAR (25)

Mandate
- **id NUMBER (8) PK**
- **FK_room_admin NUMBER (8) FK**
- **FK_room_id NUMBER (8) FK**
- date_created DATE
- deadline DATE
- description LONGTEXT

Report
- **id NUMBER (8) PK**
- **FK_computer_id NUMBER (8) FK**
- **FK_user_id NUMBER (8) FK**
- resolved NUMBER (1)
- date_created DATE
- description LONGTEXT

Location
- **id NUMBER (8) PK**
- name VARCHAR (50)
- city VARCHAR (50)

Staff_Room_Assigned
- **FK_room_admin_id NUMBER (8) PK FK**
- **FK_room_assigned_id NUMBER (8) PK FK**

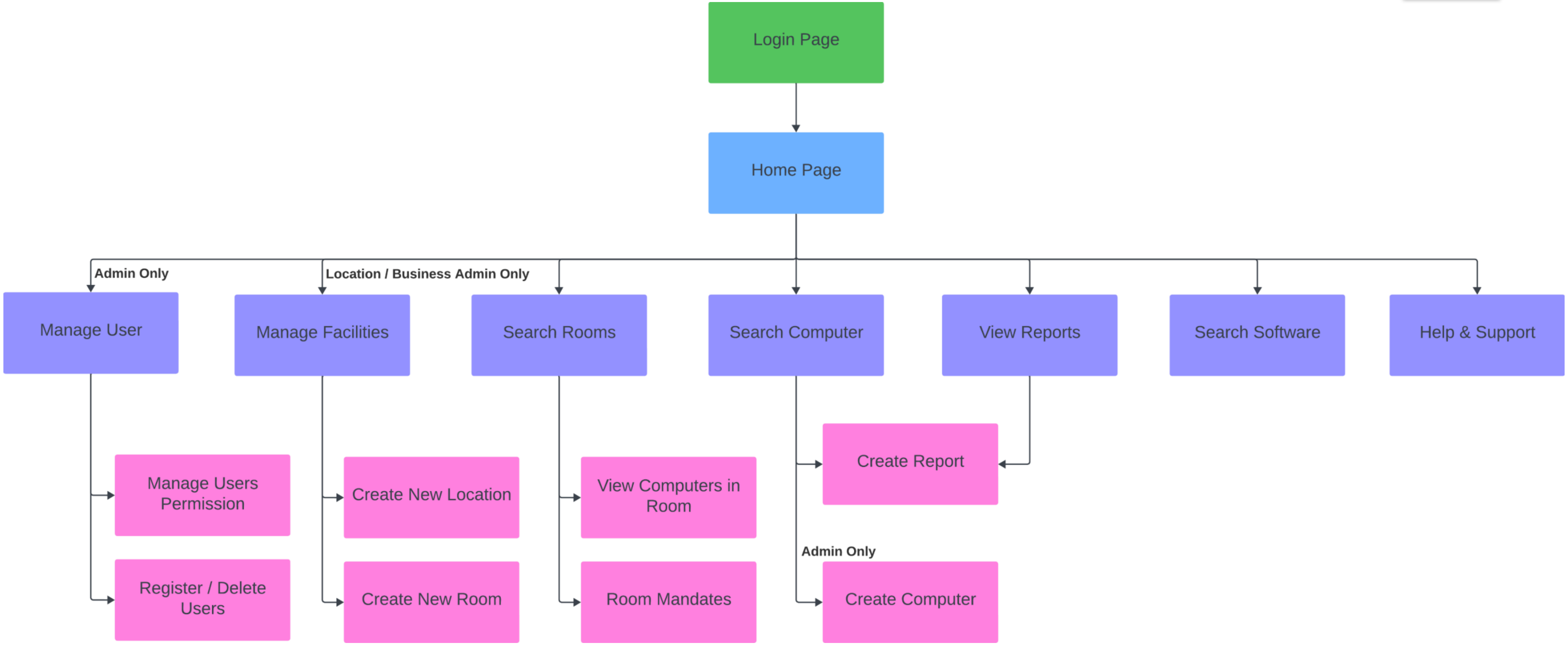
Room
- **id NUMBER (8) PK**
- **FK_location_id NUMBER (8) FK**
- name VARCHAR(100)

Computer
- **id NUMBER (8) PK**
- **FK_room_id NUMBER (8) FK**
- computer_code NUMBER (25)
- ready_for_role VARCHAR (25)

Software
- **id NUMBER (8) PK**
- name VARCHAR (100)

Program
- **id NUMBER (8) PK**
- **software_id NUMBER (8) FK**
- version VARCHAR (8)

Computer_Program
- **FK_computer_id NUMBER (8) PK FK**
- **FK_program_version_id NUMBER (8) PK FK**


Login Page
Home Page
Admin Only
Location / Business Admin Only
Manage User
Manage Facilities
Search Rooms
Search Computer
View Reports
Search Software
Help & Support
Manage Users Permission
Register / Delete Users
Create New Location
Create New Room
View Computers in Room
Room Mandates
Create Report
Admin Only
Create Computer

Scrum

- Assigned role to team members.
- Project initiation phase
- Release planning
- Sprint planning
- Daily 15 minute standups
- Sprint Review, followed by Sprint retrospective

Agile

- Iterative development
- Continuous integration
- Adaptive planning
- Cross-Functional team members

Sprint 1

Create a shell application that allows users to login and access the system

Jenny

- Collect computer information from the Sydney office computers, setting up real and dummy data for computers and programs in the database

Jefferson

- Collect computer information from the Sydney office computers, implementing navigation bar, basic tests for controllers/services

Matt

- Build back-end structure with entities, services, and controllers, and integrated JPA Inheritance for user-staff-admin relationship

Bennett

- Build back-end database, with relationships, models and services

Kelly

- Login Page, Register Page

Sprint 2

Implement creating computers and mandates. Allow users to view rooms, computers, and software. Users can also search for computers based on specific roles.

Jenny

- Restructured the programs entity relationship in the database, updating related backend classes and data, implemented display software's page, and test coverage

Jefferson

- Created the pages that allowed users to view rooms and computers inside the rooms. Worked on the computer card icon, added a welcome page.

Matt

- Implemented role-readiness to computer entities; Added Mandate functionalities

Bennett

- Developing 'Search Computers' page with implementation of 'Add computer' functionality

Kelly

- Admin Page and related backend code (staff and roomadmin)

Sprint 3

Implement issues and creating new locations and rooms. Add a timeout feature and a help me tab. Improve test coverage and usability.

Jenny

- Implemented view reports page, creating and editing reports function and test coverage

Jefferson

- Added an idle timer, and functionality to add new rooms and locations. Improved on overall useability.

Matt

- Further unit testing to check for coverage through JUnit and Mockito, bug-fixing and QOL features for computers, added Help section, refined Mandate rendering

Bennett

- Finishing the computers page and its component with a ‘Program Transfer List’, implement permission check to restrict certain functionality being used by authorized users

Kelly

- Finishing admin, allows users to filter through staff members, and bug fixes

Definition of done

- A user should be able to see what computers they can/cannot use and what software (and their versions) is installed on them.

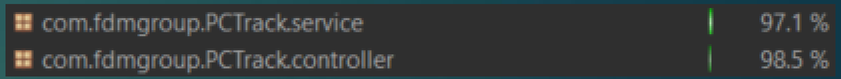
- The software should have a nice UI that users can easily interact with.

- Admins can perform their responsibilities according to their roles.

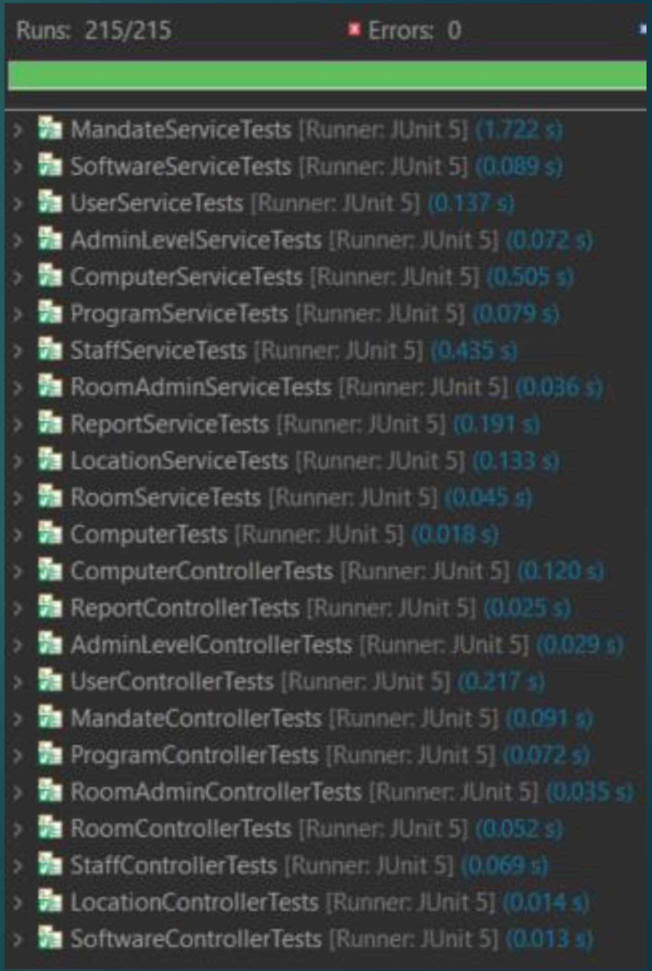
- Test coverage at least 95%

Testing

- Back-end: Unit Testing
  - Class based - Test-Driven Development
  - Utilising JUnit and Mockito module in Maven Project
  - Conducted on every new feature being implemented within the class before pushing to repository
- Back-end: Acceptance Testing
  - Service and controller functions check
  - Ensure functions meets the acceptance criteria per user story
- Back-end: Security Testing
  - Establishes security through Spring Boot security framework



```
com.fdmgroup.PCTrack.service       97.1 %
com.fdmgroup.PCTrack.controller    98.5 %
```



```
Runs: 215/215          Errors: 0

> MandateServiceTests [Runner: JUnit 5] (1.722 s)
> SoftwareServiceTests [Runner: JUnit 5] (0.089 s)
> UserServiceTests [Runner: JUnit 5] (0.137 s)
> AdminLevelServiceTests [Runner: JUnit 5] (0.072 s)
> ComputerServiceTests [Runner: JUnit 5] (0.505 s)
> ProgramServiceTests [Runner: JUnit 5] (0.079 s)
> StaffServiceTests [Runner: JUnit 5] (0.435 s)
> RoomAdminServiceTests [Runner: JUnit 5] (0.036 s)
> ReportServiceTests [Runner: JUnit 5] (0.191 s)
> LocationServiceTests [Runner: JUnit 5] (0.133 s)
> RoomServiceTests [Runner: JUnit 5] (0.045 s)
> ComputerTests [Runner: JUnit 5] (0.018 s)
> ComputerControllerTests [Runner: JUnit 5] (0.120 s)
> ReportControllerTests [Runner: JUnit 5] (0.025 s)
> AdminLevelControllerTests [Runner: JUnit 5] (0.029 s)
> UserControllerTests [Runner: JUnit 5] (0.217 s)
> MandateControllerTests [Runner: JUnit 5] (0.091 s)
> ProgramControllerTests [Runner: JUnit 5] (0.072 s)
> RoomAdminControllerTests [Runner: JUnit 5] (0.035 s)
> RoomControllerTests [Runner: JUnit 5] (0.052 s)
> StaffControllerTests [Runner: JUnit 5] (0.069 s)
> LocationControllerTests [Runner: JUnit 5] (0.014 s)
> SoftwareControllerTests [Runner: JUnit 5] (0.013 s)
```

DEMO

Reflection

Challenges

- Lots of bugs - fixing bugs last minute
- Assigning tasks back and forth – tasks has been swapped between people within sprint

What went well

- Our simplistic and modern web design looks very clean, navigation around the site is simple along with a 'Help page' to support any new users
- Displaying computer cards and have it update its colour according to its role ready status allows users to quickly determine the programs without looking into the programs list

Future Development Ideas

- Implementing “Request” ticket system to higher-level staff for Users
  - Change request from users that can only be access by higher-ups
- Notifications and Alert System
  - Regarding on computers issue based on the location of the admin
- Forget Password System
- Summary Report
  - Display computer performance / usage, categorized based on role + location
- Software / Program Database Editing
  - Implementing add, edit and delete functionalities
- Able to edit presets
  - Version comparison and check system implemented

Questions?