CS3733-D18-D02 Project BW3D Iteration 4
Software Engineering - Prof. Wong

Worcester Polytechnic Institute
Computer Science Department

Team

M - Manly Minotaurs

Team Coach

Ebenezer Ampiah

Team Roles

Team Member	Primary Role	Secondary Role
Christian Cedron	Lead Software Engineer	
Andrew Morrison	Assistant Lead Software Engineer	
Jared Grimm	Assistant Lead Software Engineer	
Junbong Jang	Assistant Lead Software Engineer	
Joseph Yuen	Software Engineer	Project Manager
Zachary Emil	Software Engineer	Product Owner
Peter Christakos	Software Engineer	Scrum Master
Oliver Sanderson	Software Engineer	Test Engineer
Andre Imperiali	Software Engineer	Documentation Analyst
Rohit Unnam	Software Engineer	

I. Project Overview

The Brigham and Women's Hospital located in Boston, MA is "an international leader in virtually every area of medicine and has been the site of pioneering breakthroughs that have improved lives around the world." While Brigham and Women's is at the forefront of medical breakthrough, the hospital, like many hospitals across the U.S, has an outdated navigation system that may be confusing to the large number of visitors and patients who come to the medical establishment. Our project is a simple software solution to conveniently help visitors, patients, and staff easily navigate through the complexity of Brigham and Women's layout. In addition to its navigation functionality, the software also includes relevant service request capabilities and other convenient features for a better user experience. Some of those features include staff requests, item room requests, a gift shop browser experience, an emergency server system, the kiosk log, and access to the existing hospital patient portal. In addition to our main kiosk application, we produced an admin web application so that admins can utilize the admin functionality of the kiosk on their mobile devices.

II. Software Development Environment

Function	Application
IDE	IntelliJ 2017.3.5 (Ultimate Edition)
Build	Gradle, TravisCI
Language	Java JDK SE 8
UI	JavaFX with Scenebuilder 8
Database	Java DB (Apache Derby)
Diagrams	Lucid Chart
Testing	JUnit, Integration Testing
Repository	GitHub
Project Management	GitHub Project
Issue Tracking	GitHub Issues
Communications	Slack
External Jar Library	fontawesomefx-8.9.jar
External Jar Library	controlsfx-8.40.14.jar
External Jar Library	jSerialComm-2.0.2.jar
External Jar Library	RXTXcomm.jar
External Jar Library	Jfoenix 8.0.1 jar

III. User Stories

Iteration 1

- As a visitor/patient/admin, I want to find a path from a starting location to a destination, so that I know how to get to a destination.
- As a patient, I want to request for a nurse so that I can get help.
- As an admin, I want to know if a patient requests for something so that I can send help.
- As a service provider, I want to enter my unique id onto a kiosk, so that I can confirm that I completed a service request.
- As an admin, I want to add, remove, and edit nodes, so that the map contains the most up to date information.
- As an admin, I want to geofence, so that patients and visitors avoid hazards and restricted areas.
- As an admin, I want to add a custom destination based on coordinates, so I can go to destinations that are not nodes.
- As a visitor/patient/admin, I want to receive textual step-by-step directions to my end destination, so that I can get there easier.
- As a visitor/patient/admin, I want to be able to request a translator, so that I can communicate with others.
- As a visitor/patient/admin, I want to notify for a clean-up so that there's no hazards.
- As a visitor/patient/admin, I want to know ETA and time to get there so that I can plan my trip
- As a visitor/patient/admin, I want to have quick directions to the closest bathrooms/utilities and food, so that I can get access to what I need quickly.
- As a visitor/patient/admin, I want to choose a handicap accessible path, so I can get where I need to go with ease.
- As a visitor/patient/admin, I want to add waypoints along my path, so I can go to multiple locations without having to stop at multiple kiosks.
- As a visitor/patient/admin, I want to choose a language so that I can understand the directions.
- As a visitor/patient/admin, I want to see detailed map info, so that I can specify amenities.
- As a visitor/patient/admin, I want to find the nearest exit, so that I can leave the building quickly during an emergency.
- As a visitor/patient/admin, I want to see an event schedule so that I can know what's going on.
- As an admin, I want to add events so that I can notify patients and visitors.
- As a patient, I want to view my profile information, so I can get directions to my room.
- As an admin, I want to view my patients' information, so I can get directions to their rooms.

- As an admin, I want to see Kiosk history so that I can know where my patients have been.

Iteration 2

- As a employee/admin or a visitor/patient in an emergency, I want to notify for a clean-up so that there's no hazards.
- As a visitor/patient/admin, I want to have quick directions to the closest bathrooms/utilities and food, so that I can get access to what I need quickly.
- As a visitor/patient/admin, I want to find a path from a starting location to a destination, so that I know how to get to a destination.
- As a patient, I want to request for a nurse so that I can get help.
- As an admin, I want to add, remove, and edit nodes, so that the map contains the most up to date information.
- As a visitor/patient/admin, I want to receive textual step-by-step directions to my end destination, so that I can get there easier.
- As an admin, I want to be able to select the pathfinding algorithm used
- As an admin, I want to be able to add, remove, and modify employee information
- As an admin, I want to be able to track records of service providers

Iteration 3

- As a employee/admin or a visitor/patient in an emergency, I want to be able to request a translator, so that I can communicate with others.
- As a visitor/patient/admin, I want to know ETA and time to get there so that I can plan my trip
- As a visitor/patient/admin, I want to see detailed map info, so that I can specify amenities.
- As a visitor/patient/admin, I want to find the nearest exit, so that I can leave the building quickly during an emergency.
- As an admin, I want to see Kiosk history so that I can know where my patients have been.
- As an admin, I want to be able to select the pathfinding algorithm used
- As a visitor/patient/admin, I want to find a path from a starting location to a destination, so that I know how to get to a destination.
- As a visitor/patient/admin, I want to receive textual step-by-step directions to my end destination, so that I can get there easier.
- As an admin, I want to add, remove, and edit nodes, so that the map contains the most up to date information.
- As a user, I want to be able to manipulate my view of the map, so that I can have a better view of the map
- As a user, I want to be able to receive directions on my personal device, so that I don't have to view my directions on a kiosk all the time

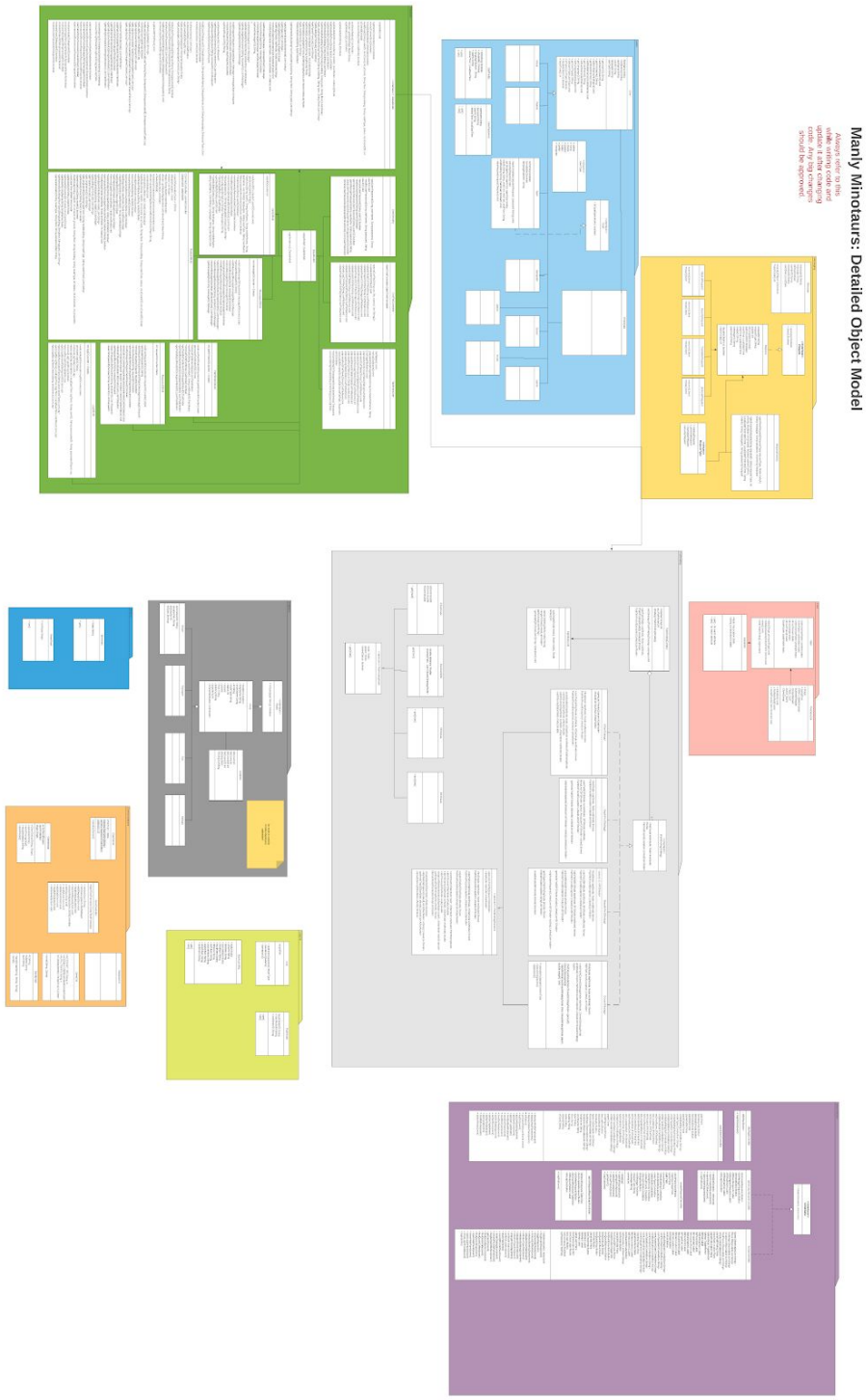
- As a user, I want to have a material design UI, so that I can easily navigate the program
- As a user, I want autocomplete search, so that I don't have to know the exact name of a location
- As a user, I want to view my floor progression, so that I can have a good sense of moving to different floors
- As a patient/employee/admin, I want an automatic fire detector, so that the system automatically goes into emergency mode when there's a fire
- As a patient/employee/admin, I want all kiosks to be set to emergency mode when one is, so that everyone is aware of an emergency
- As an Admin, I want to be able to undo changes made to users, so that I can revert mistakes
- As a user, I want to access the B & W website portal, so that I can access my existing account

Iteration 4

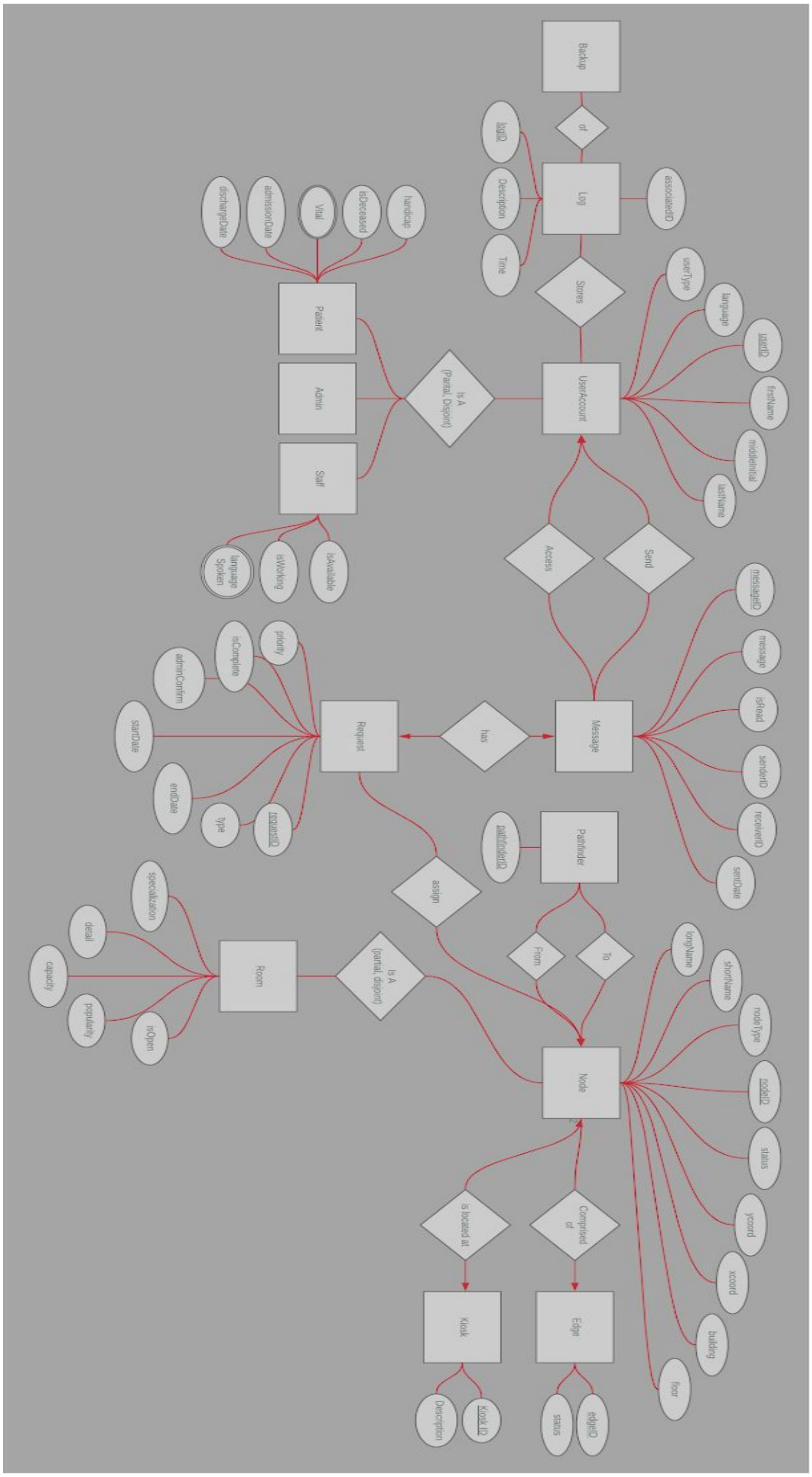
- As an admin, I want to be able to set an idle time, so that new users are greeted by the landing screen
- As a user, I want to be able to manipulate my view of the map, so that I can have a better view of the map
- As a patient/employee/admin, I want an automatic fire detector, so that the system automatically goes into emergency mode when there's a fire
- As an admin, I want to add, remove, and edit nodes, so that the map contains the most up to date information.
- As a patient, I want to make room service requests, so that I can get what I need
- As a user, I want to be able to select items to buy from the gift shop, so that I can see items and prices easily
- As a user, I want to view my floor progression, so that I can have a good sense of moving to different floors
- As a visitor/patient/admin, I want to receive textual step-by-step directions to my end destination, so that I can get there easier.
- As a user, I want autocomplete search, so that I don't have to know the exact name of a location
- As a user, I want to have a material design UI, so that I can easily navigate the program
- As a user, I want an animated map for pathfinding, so that I can more easily visualize my path
- As a User, I want a welcome/idle screen so that I am not overwhelmed when I approach the kiosk
- As a visitor/patient/admin, I want to find a path from a starting location to a destination, so that I know how to get to a destination.
- As a user, I want a tutorial, so that I know how to properly use the application

IV. Diagrams

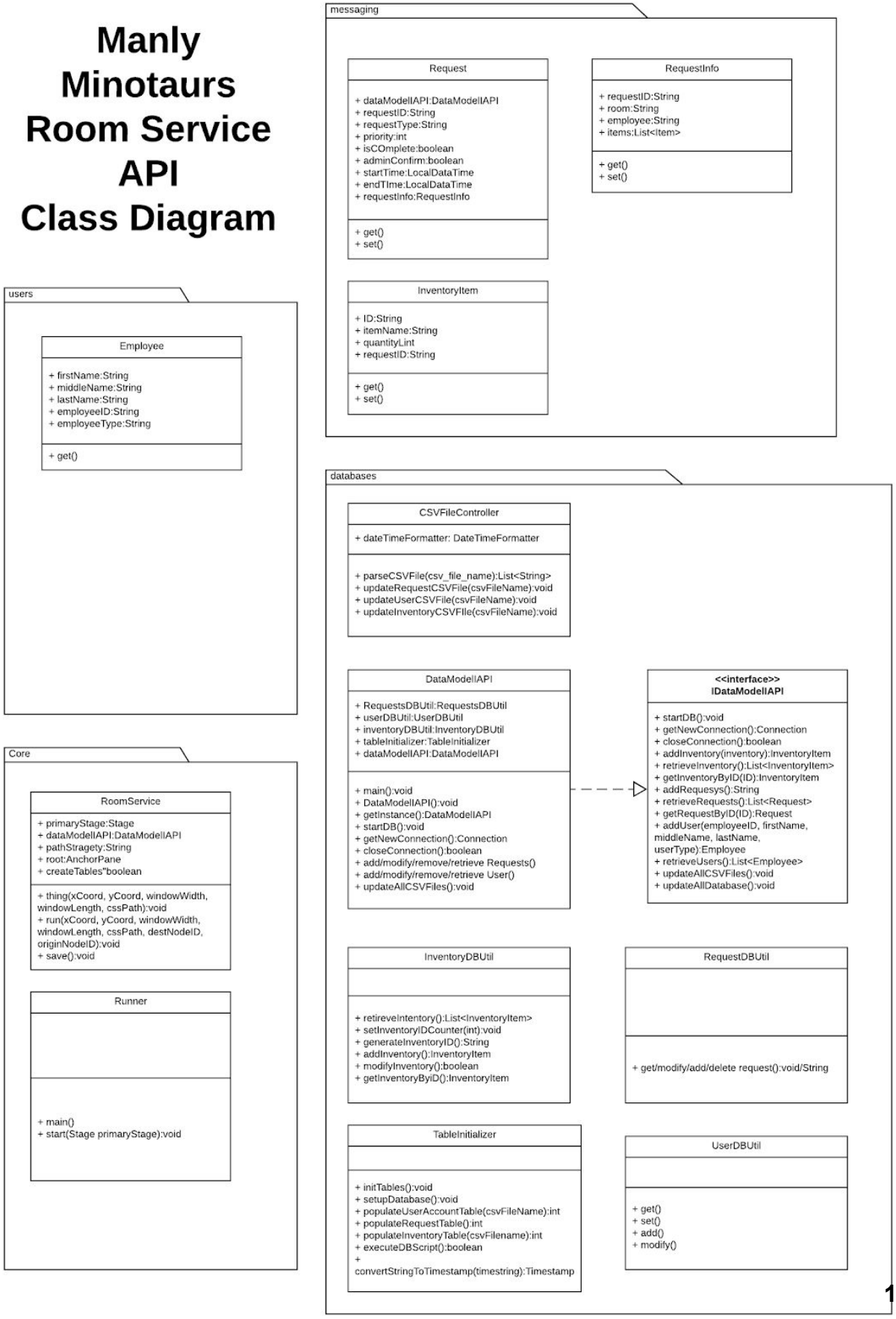
Class Diagram:



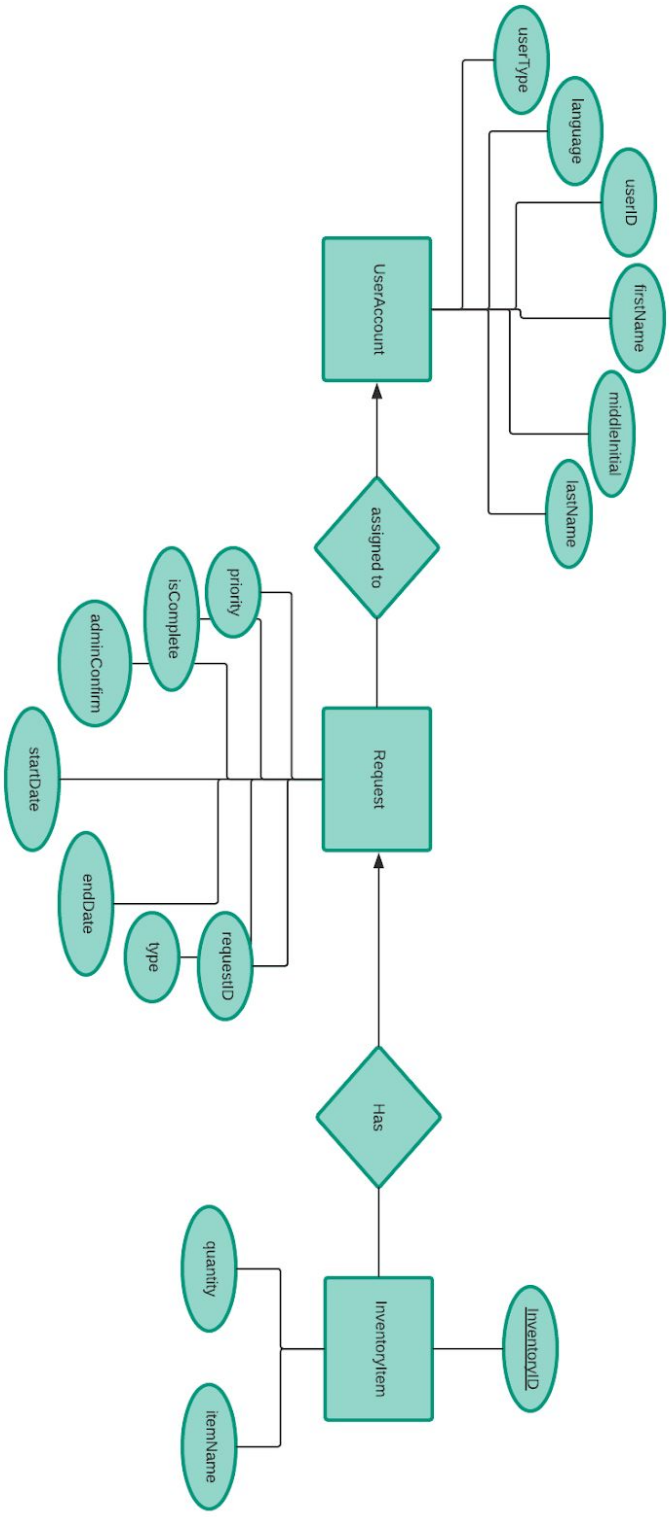
ERD Diagram:


Backup
of
logID
Log
Description
Time
associatedID
Stores
UserAccount
userType
language
userID
firstName
middleInitial
lastName
Is A
(Partial, Disjoint)
Patient
Admin
Staff
dischargeDate
admissionDate
Vital
isDeceased
handicap
language Spoken
isWorking
isAvailable
Access
Send
Message
messageID
message
isRead
senderID
receiverID
sentDate
has
Request
priority
isComplete
adminConfirm
startDate
endDate
type
requestID
assign
Pathfinder
pathfinderID
From
To
Room
specialization
detail
capacity
popularity
isOpen
Is A
(partial, disjoint)
Node
longName
shortName
nodeType
nodeID
status
ycoord
xcoord
building
floor
is located at
Comprised of
Kiosk
Edge
Kiosk ID
Description
edgeID
status

Manly Minotaurs Room Service API Class Diagram



API ERD Diagram:



IV. Design Patterns

Proxy - ProxyImage.java
ProxyImage was used mainly to load the map images. Before the implementation of the pattern, we had to set the images manually to a file and then link it to an imageview which took up a considerable amount of code. After the implementation of the proxy image, we could put simply the name of the map image and the imageview and set the map in just one line.

Momento - IdleMapController.java
Upon entering the IdleMapController, the idle state is restored. Upon leaving it, the idle state is saved. This allows us to logout people who are idle and does not save their progress. Essentially, this allows us to protect against partially entered information.

Observer - Most tables used in the project
While we didn't explicitly write an observer pattern, we utilize Java's built in observable lists that utilize the observer pattern for our tables. When an item is added or removed from an observable list, the list updates automatically. Our clock on the launch page of the application also utilizes a Java object that uses the observer pattern.

Singleton - DataModelI.java
DataModelI, in addition to being a facade, was made a singleton since multiple instances of this class will never be necessary. Its sole purpose is to allow interactions with the database. This makes things easy since one can always call DataModelI.getInstance().*util name*.*function name* instead of making their own instance of DataModelI, which would be unnecessary.

Facade Pattern - DataModelI.java
Our database package uses DataModelI as a facade to abstract away all code that interacts with the database. This allows other packages to easily interact with the database. Our code is simplified even more since all packages simply use DataModelI instead of many individual database utility classes.

Strategy Pattern - IPathfindingStrategy.java
In IPathfindingStrategy, there is a function called getPath which takes in a start node, end node, and a pathfinding strategy such as A*, Dijkstra's, Breadth first search or depth first search. Each of these functions then run their own function called find which is a wrapper function for their calcPath that can take in their own node classes with custom fields.

Template Pattern - DistancePathFinder.java
DistancePathfinder is an abstract class which includes many distance-based functions used by both A* and Dijkstra's algorithm. A* and Dijkstras then extend the abstract class in order to use its functions.

V. Programmer Responsibilities

Coding Responsibilities

Team Member	Responsibility
Christian Cedron	messaging system to send a nurse request to admin for authorization (iter1)
	service request
	cleanup request
	system to convert a pixel on the map to actual distance in the application option singleton
	system to send text messages
	system to send emails
	server-client system
	function to reset application to non-emergency state
	Return the application to the initial pathfinding screen after set idle time, using memento pattern
	UI option allows Admin to set value to idle time
	Saves custom time-out time between application runs
	database ported to web app
	real-time clock
Andrew Morrison	database for requests (iter1)
	database for unique service provider IDs (iter1)
	database for translators, and what languages they can translate
	Add and remove functions for translators, languages spoken
	Scenebuilder layout to allow entering of node coordinates by clicking on the map
	Pathfinding algorithm to find quickest and safest paths during an emergency (iter3)
	Hash table for database
	UI zoom and rotate the map view
	UI floor button (iter3)
	Grey out floors not in focus (iter3)
	General animations in UI

	zoom slider
	map UI
	Smooth transistion
	animated arrow that follows direction of path
	Breadcrumbs in pathfinding
	Automatically snap map center to pathfinding
	UI indication of direction of path
Jared Grimm	Scenebuilder Layout for Main Pathfinding (iter1)
	Main Request Button (iter1)
	Scenebuilder layout for making requests (iter1)
	Scenebuilder layout for admin to view and manage requests (iter1)
	Scenebuilder layout for contacting help (iter1)
	Scenebuilder ID input option (iter1)
	Scenebuilder node editing ui (iter1)
	text-based option button (iter1)
	Scenebuilder layout for translator requests (iter2)
	Scenebuilder request option for clean-up and location of hazard (iter2)
	Scenebuilder interface to select directions to bathrooms/utilities/food (iter2)
	Scenebuilder option for handicap accessible mode(iter2)
	Scenebuilder login layout (iter2)
	Scenebuilder admin login layout (iter2)
	Scenebuilder layout to view kiosk history (iter2)
	Scenebuilder layout for both 2D and 3D map, and the option to select one or the other.
	Scenebuilder layout to show multiple floors (iter2)
	Scenebuilder layout to allow admin to select algorithm (iter2)
	Scenebuilder layout to allow admin to view and modify employees
	Scenebuilder layout to enter report information (iter2)

	Scenebuilder layout for translator requests (iter3)
	Scenebuilder emergency button (iter3)
	Scenebuilder special emergency layout (iter3)
	Scenebuilder layout to view kiosk history
	turn icons in UI to accompany text
	UI context menu to choose between click-and-drag, adding new nodes, and creating edges (iter3)
	UI to click-and-drag nodes to modify their location (iter3)
	edges in node editor UI (iter3)
	Updated UI map icons (iter3)
	Updated UI map buttons (iter3)
	Updated UI panels (iter3)
	Updated UI dashboard (iter3)
	emergency UI screen (iter3)
	undo button to user add/remove/modify UI
	BW Website connected to application
	UI layout to interact with website portal
	UI option to access and close the website portal
	Emergency screen
	emergency voice readout
	Edges follow nodes while dragged
	node editor UI
	Impliment another team's API
	heirarchial text directions
Junbong Jang	database for requests (iter1)
	database for unique service provider IDs (iter1)
	Create a table for janitors
	Add and remove functions for janitor database
	database to track report information for services provided.
	database table for translators, and what languages they can translate (iter3)
	Add and remove functions for translators,

	languages spoken (iter3)
	Logging class and controller functions for Kiosk History (iter3)
	database to store kiosk history
	Database to store path finding history
	Pathfinding Controller class for accessing path finding history
	Hash table for database
	Facade pattern to simplify structure
	database for service requests
	database to store all changes made to users database
	database function to add changes when changes are made, and remove changes when change is undone
	adding requests to database
Joseph Yuen	Scenebuilder Layout for Main Pathfinding (iter1)
	Main Request Button (iter1)
	Scenebuilder layout for making requests (iter1)
	Scenebuilder layout for admin to view and manage requests (iter1)
	Scenebuilder layout for contacting help (iter1)
	Scenebuilder ID input option (iter1)
	Scenebuilder node editing ui (iter1)
	text-based option button (iter1)
	Scenebuilder layout for translator requests (iter2)
	Scenebuilder request option for clean-up and location of hazard (iter2)
	Scenebuilder interface to select directions to bathrooms/utilities/food (iter2)
	Scenebuilder option for handicap accessible mode(iter2)
	Scenebuilder login layout (iter2)
	Scenebuilder admin login layout (iter2)
	Scenebuilder layout to view kiosk history (iter2)
	Proxy design pattern for the loading map images

	(iter2)
	Scenebuilder layout for both 2D and 3D map, and the option to select one or the other.
	Scenebuilder layout to show multiple floors (iter2)
	Scenebuilder layout to allow admin to select algorithm (iter2)
	Scenebuilder layout to allow admin to view and modify employees
	Scenebuilder layout to enter report information (iter2)
	UI of a list of destination a user may select (iter3)
	UI to show users how multiple floor directions are navigated (iter3)
	UI zoom and rotate the map view
	UI floor button (iter3)
	UI option to select and enter in email and/or phone information
	Updated UI map icons (iter3)
	Updated UI map buttons (iter3)
	Updated UI panels (iter3)
	Updated UI dashboard (iter3)
	autocomplete dropdown for UI search boxes (allow user to select location by clicking on menu)
	service that predicts what user will type
	compass to UI
	Create drop-down list of possible items, rather than free text entry for API
	Increase text size for API
	text fuzzy search
	search UI
	main pathfinding UI
	admin interface UI
	Breadcrumbs in pathfinding
	Idle map UI
	About and Credit page
	tutorial in the UI

Zachary Emil	system to convert a pixel on the map to actual distance in the application option singleton
	node labels follow rotated map
	About and Credit page
	real-time clock
Peter Christakos	A* pathfinding
	Scenebuilder node editing ui (iter1)
	A* to finds best path between different floors
	depth-first and breadth-first algorithm (Use the nested private class Singleton design pattern for algorithm selection) (iter2)
	Pathfinding algorithm to find quickest and safest paths during an emergency (iter3)
	Djikstra’s algorithm and bredth first search Using template pattern in combination with strategy pattern
	Djikstra's algorithm
	system to convert a pixel on the map to actual distance in the application option singleton
	turns other than a simple left or right
	UI option allows Admin to set value to idle time
	heirarchial text directions
	Idle map UI
	closest pathfinding algorithm
Oliver Sanderson	A* pathfinding
	Human readable direction from A* (iter1)
	A* to finds best path between different floors
	depth-first and breadth-first algorithm (Use the nested private class Singleton design pattern for algorithm selection) (iter2)
	Testing using TravisCI
	fire detecting device
	fire detection software integrated into application emergency system
	application ported to web app

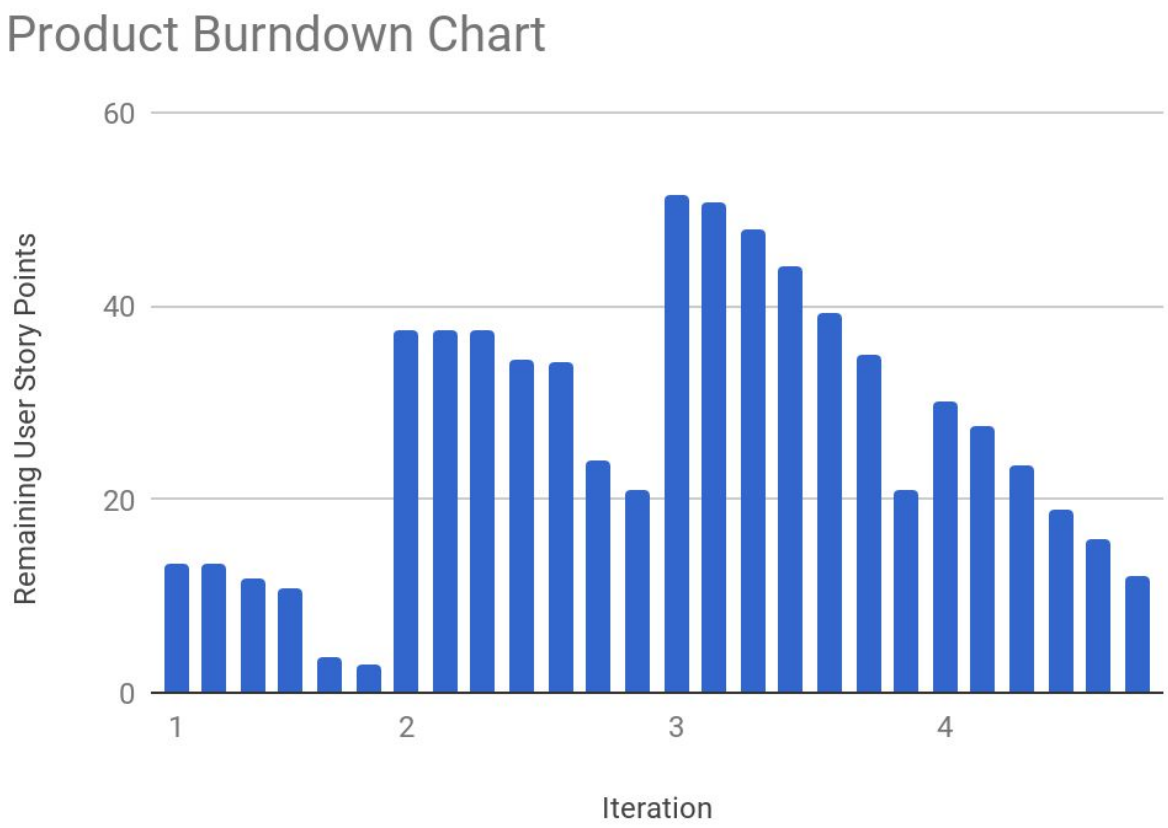
Andre Imperiali	Human readable direction from A* (iter1)
	database with human-readable names for nodes
	database table for translators, and what languages they can translate (iter3)
	Add and remove functions for translators, languages spoken (iter3)
	Logging class and controller functions for Kiosk History (iter3)
	Pathfinding Controller class for accessing path finding history
	database for service requests
	adding requests to database
	About and Credit page
Rohit Unnam	scenebuilder layout for 2-d map (iter1)
	scenebuilder layout for 3-d map (iter1)
	database for unique service provider IDs (iter1)
	Scenebuilder node editing ui (iter1)
	Scenebuilder layout for translator requests (iter2)
	Scenebuilder request option for clean-up and location of hazard (iter2)
	Scenebuilder interface to select directions to bathrooms/utilities/food (iter2)
	Scenebuilder option for handicap accessible mode(iter2)
	Scenebuilder login layout (iter2)
	Scenebuilder admin login layout (iter2)
	Scenebuilder layout to view kiosk history (iter2)
	Scenebuilder layout for both 2D and 3D map, and the option to select one or the other.
	Scenebuilder layout to show multiple floors (iter2)
	Scenebuilder layout to allow admin to select algorithm (iter2)
	Scenebuilder layout to allow admin to view and modify employees
	Scenebuilder layout to enter report information

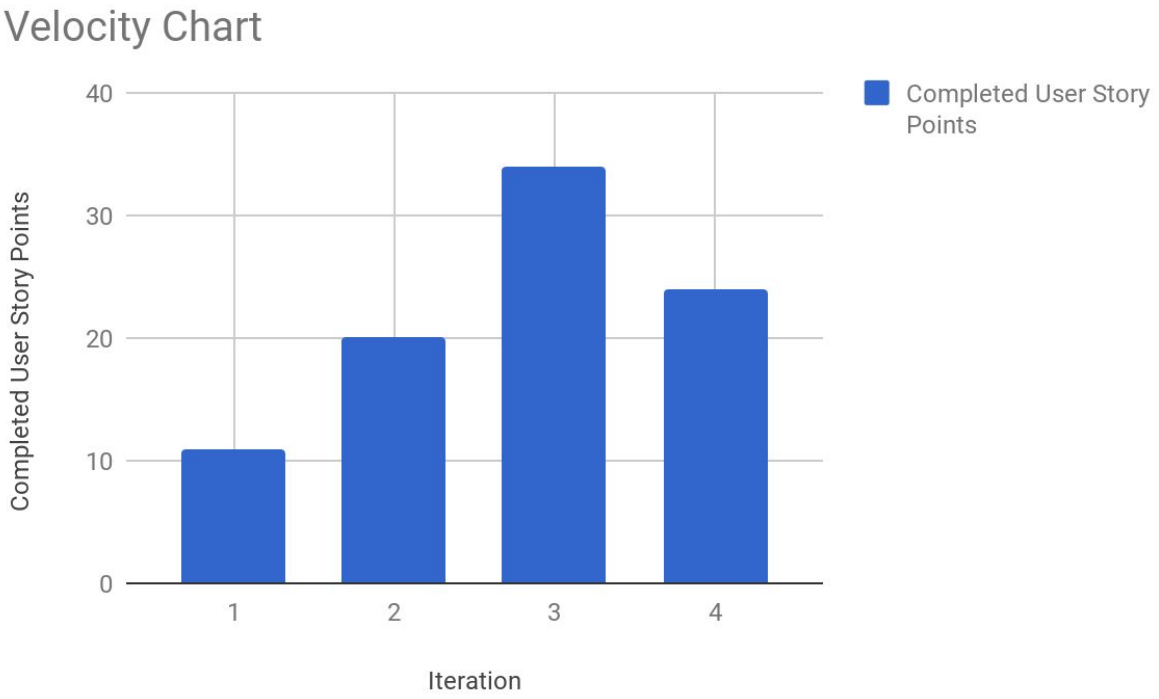
	(iter2)
	Updated UI panels (iter3)

Non Coding Responsibilities

Christian Cedron	advising and assisting other software devs (all four iterations)
Peter Christakos	SCRUM master; logged SCRUM meetings (all four iterations)
Zachary Emil	product owner; wrote tasks/user stories and burndown charts (all four iterations)
Andre Imperiali	document analyst; JavaDoc documentation, UML diagram (all four iterations)
Joseph Yuen	project manager; delegated tasks, wrote much of final deliverable doc (all four iterations)

VI. Product Burndown





Iteration 4 Average Velocity: 3.833 User points per day

VII. Team Reflection

Part 1: What did the team learn about software designing through iterations? Did your later designs better match the subsequent code written or not? Discuss why.

The team learned several lessons about software designing throughout iterations one through four. For starters, the team learned to ALWAYS allot more time for each task. In addition, the team learned to sprint plan on Friday, refactor friday night, and start tasks as early as possible. Starting the bulk of an iteration on Monday was never a good idea. Planning and designing was far more important than originally anticipated and subsequent code very closely matched later designs for iterations three and four. This is because the team realized that diverging from original plans would cause confusion and miscommunication through other team members and would make debugging a nightmare.

Part 2: What did the team learn about software methodologies and working together as a team throughout the iterations? Did the team operate better or worse as time progressed? Why?

The team learned that implementing good software methodologies is far more important than previously anticipated. The most difficult part of software engineering is the collaboration. Being able to effectively utilize the potential of ten people is the most valuable skill a software engineering team can have. The Manly Minotaurs struggled in achieving this for iterations one and two. In order to properly delegate responsibilities, each member needed to be more comfortable and more familiar with their teammates. As time progressed, this proved to be true. The more cohesive the team became, the more efficient the work became.

Part 3: What did your team do to improve team cohesion and foster a team culture?

To improve team cohesion, the team made a large effort in taking a small portion of meeting times to relax and socialize. They also would go out often to get meals and learn more about each other. As the iterations progressed and each member became more comfortable with each other, the team cohesion improved and the team efficiently also increased proportionally.

Part 4: What would the team have done differently in hindsight? What did the team learn as a result of unexpected challenges? What advice would you give future teams taking CS 3733 to improve their experience?

In hindsight, the team would’ve made a larger effort to get to know each other during the first three weeks of class when iterations had not yet been assigned. In addition, the team should’ve started working on the homework assignments earlier because the programming tasks would always take longer than the expected completion time. As a result of unexpected challenges, the team learned that their biggest assets are each other. Having 9 other teammates with different backgrounds working on the same project made solving any problem virtually possible. Advice for a future Software Engineering team would be to build a close relationship with their teammates, communicate often with their team coach, and get on top of tasks as soon as possible.

Project Management

Team Position Changes

- No changes to report

Salary Distribution

Method

For being a member of the team, attending team meetings, and participating in team discussion, team members are awarded a base salary of 3 WPI dollars. In order to maintain a positive team environment, WPI dollars are never deducted from a team member except in extreme circumstances. 150 dollars are awarded based on the weight of tasks in the product backlog. If a team member does an outstanding job and the whole team acknowledges a team member's accomplishment, extra WPI dollars may be awarded at the discretion of the Project Manager. Any remain WPI dollars at the end of an iteration or project will be allocated equally to all team members. In the rationale portion of the chart, tasks completed are described, and their weights are listed.

Last Name	First Name	Salary	Rationale
Cedron	Christian	23.83	Christian set up the memento pattern for returning the application to the launch screen and helped to port the database to the web application (+17.5) and helped other teammates with their tasks (+3) (+.33).
Christakos	Peter	18.83	Peter edited the nearest navigation functionality, drafted the launch screen UI, and processed some of the directions data for hierarchical text directions (+12.5) and performed scrum and documentation duties (+3) (+.33).
Emil	Zachary	13.21	Zachary rotated node labels so that they appeared at the right angle and helped to create the real-time clock (+6.88) and performed product owner responsibilities (+3) (+.33).
Grimm	Jared	28.83	Jared touched up the node editor, improved the emergency screen, helped to generate hierarchical text directions, and assisted in making the web application (+25) and assisted other developers with their tasks (+0.5) (+.33).

Imperiali	Andre	11.30	Andre fixed adding requests to the database and smoothed out the bugs for the API (+5.94) and performed documentation analyst duties (+2) (.36).
Jang	Junbong	23.39	Junbong fixed up the API's database and constructed the database for the web application (+14.06) and assisted other coders in their assigned tasks (+4) (+2) (+.33).
Morrison	Andrew	28.21	Andrew created new map functionality and fixed many of the bugs for a better user experience (+24.38) and helped other developers with their code (+0.5) (+0.33).
Sanderson	Oliver	26.39	Oliver headed and constructed the main web application and fixed up pathfinding (+19.06) and assisted other developers in their tasks (+4) (+0.33).
Unnam	Rohit	0.00	Rohit did not attend major meetings (-1), did not deliver assigned tasks (-1), and performed tasks he was explicitly asked not to do (-1)
Yuen	Joseph	26.01	Joseph redesigned the navigation screen UI, constructed many of the map features and functionality, and performed project manager duties (+24.68) (+.33) and gave away two points to Junbong for his efforts to make a new database in such short time (-2).