Team 13 Product Backlog
BoilerCheck

Team Members:
Kinshuk Juneja, Nadeem Mahmood, Jeremiah Murphy, Michael Reed, Jacob Richwine, Zhandos Suleimenov

Problem Statement:
As the enrollment at Purdue continues to increase, it is becoming harder to find a non-crowded place to meet with your group, study for classes, or a dining court that isn't packed. Existing programs only cover labs or other subsets of buildings; ours will encompass multiple categories of buildings. Our program, the BoilerCheck Android app, will allow users to check-in and report conditions at popular buildings around campus. Users will also be able to view current statistics of the building before making a decision on where to go.

Background Information:
There is currently a market gap that our product will fill. We could not find a current similar tool that solves the problem that ours sets out to do. Our app solves the major issue of overcrowding on campus and remedies this by providing instant feedback to users about the status of buildings around campus, as well as allow users to choose where to go with more information at hand. This allows users to find a quiet study place, a non-crowded eatery, or a non-packed time to go workout at the CoRec. Not only will the app detect your current whereabouts, it will direct you to the nearest facility that you desire.

Environment:
We will be developing for Android with Android Studio in IntelliJ's IDEA. To handle requests and deliver responses to the app we will be using Node.js as a simple REST API. To store our data we will be using a simple persistent NoSQL Database based on MongoDB called NEdb. We have chosen Android because the majority of the team has Android phones and is familiar with Java over Objective-C. Node.js is preferred due to its ease of use, availability, and familiarity with several teammates.

Requirements:

Functional (~200 Total Hours):

1. As a user, I want to be choose between Dining Courts, Recreation or Study Locations. (Total Hours(17): Android Learning (5) + Button Design (2) + User Interaction (5) + Multi-pages (5))
2. As a user, I want to be able to check-in based on GPS. (Total Hours(42): Learning Google Maps API (10) + Adding Building Entries to Geo-Fence (10) + Determining Buildings (2) + Designing Interface (5) + Database Creation (10) + API for App (5)
3. As a user, I want to be able to filter the list of buildings based on capacity and distance (Total Hours (40): Build GUI (10) + Database Tables (10) + API to retrieve data (5) + Maps API Distance Refresh (10) + Retrieve Building Capacities (5)
4. As a user, I want to be able to see the status of the buildings (e.g. Total Capacity, Current Capacity, Distance). (Total Hours(Same as Above Story)
5. As a user, I want to be able to see the stats of the buildings (e.g. Average/Peak Capacity). (If time allows) (Total Hours(Same as Above Story)
6. As a user, I want to be automatically checked out when I leave (Total Hours (20): Google Maps API(5) + GeoFencing Detection (10) + Update Database (5)
7. As a user, I want to receive a notification when I check in, or am detected leaving a building(Total Hours (10)
8. As a user, I want to report any bugs or errors (Total Hours (15): GUI(5) + API (5) + Notify Developers (5))
9. As a user, I want to register/create an account so my information is saved between devices(Total Hours (45): Research Authentication (10) + Create GUI (10) + API (5) + Database Handling (5) + Implement Authentication Library (10) + Creation Email (5)
10. As a user I want to login into my account. (Total Hours (10): Authentication Code (5) + Token Handling (5)
11. As a user, I want to choose between creating an account or logging in(Total Hours: GUI 2)
12. As an admin, I want to reward users with incentives(Hours: TDB). (if times allows)

Non-Functional:

1. We must be able run this mobile application on Android.
2. The application's GUI must be responsive.
3. The interface needs to be simple and user-friendly.
4. Must only allow Purdue affiliates to use the app.
5. As a developer, I want to utilize existing Google Maps API for geolocation.
6. As a developer, I want to easily add new buildings to the existing categories.
7. As a developer, I want to receive bug reports or new building requests from users.
8. As a user, I would like to have fast response times and efficient use of bandwidth.
9. The consequences of security breaches are low, and therefore the only security precautions we will take are data encryption and connection encryption between client and server.
10. The connections between the UI, server and database will be secure.

11. The server must be able to handle 10+ simultaneous requests.

Use Cases:

Case: Create an account	**System Responses**
1. Launch the app	2. The login page appears
3. Click create an account below the login	4. The create an account page shows up
5. Enter your purdue.edu email in the email box	
6. Enter your desired password in the password box	
7. Enter the same password in the confirm password box	
8. Click create an account page	9. The user is taken to the main with a dialog box confirming their account creation

Case: Login	**System Responses:**
1. Launch the app	2. The login page appears
3. Enter your purdue.edu email in the email box	
4. Enter your password in the password box	
5. Press the Login button	6. The main page appears with a dialog box saying login successful

Case: Logout	**System Responses:**
1. Click the logout button	2. The user is logged out, their location is removed from the app and they are taken to the login page.

Case: Report Bugs	**System response**
1. Log in using login and password	2. Home page appears.
3. At home page, click on the settings icon on the upper right corner.	4. List of things to choose from appears.
5. Click on “Report Bugs”	6. The report dialog appears
7. Type any error encountered.	9. Message “Your Report Has Been Sent” appears.
8. Click “send” button.	

Case: Filter list of buildings	**System Responses**
1. Click on the category of buildings you would like to see	2. The list of requested buildings appear
3. Click filter button on top right of the screen	4. The list is filtered using your filter choice

Case: Notification for change of status	**System Responses**
1. Click on the category of buildings you would like to see	2. List of buildings appear
3. Select the building you want to check into	4. A page with info about the building appears and a check-in button
5. Click the check-in button	6. Confirmation box appears
7. Click confirm	8. Box stating check-in successful appears
9. Exit the app, and walk to the building	10. A notification stating you have been checked-in appears

Case: Find and check into nearby spot	**System Responses**
1. Launch app	2. Display the three location categories
3. Tap desired category	4. Display list of buildings in desired category with quick capacity/distance stats
5. Tap filter by distance	6. Filter list of building by distance and show quick capacity/distance stats
7. Tap a building entry of interest	8. Expand building listing to show more detailed information

9. Tap Check-in	10. Display check-in registered and prompt user to hurry to location

Case: Manually check-in current location	**System Responses**
1. Launch app	2. Display the three location categories
	3. Inquire in dialog if user wishes to check-in and display best guess of current location
4. Confirm dialog	5. Display check-in successful

Case: See the current status of Campus Buildings	**System Responses**
1. Click on the category you wish to see(Dining, Study, Recreation)	2. Display Buildings in that category User
3. Click the specific building	4. Display current statistics relating to that building

Case: Auto Check out user	**System Responses**
1. While being checked-in into the app, send location to server	2. Contact Google API with current GPS coords
	3. Detect if user has left the geo-fence that they were checked in
	4. If left area of current building, send checkout notice to user's app
	5. Update database of building user was in
	6. Display message/notification notifying user he has left the current building