

Learning Labyrinth

Requirements and Specifications Document

2023-12-14, Version 2.1

Project Abstract

Learning Labyrinth is an educational aid application for beginner Java programmers that challenges them to write Java code that will enable a simulated robot to escape a simple two-dimensional maze in as few moves as possible. The application tests the user's written code and shows the resulting movement of the robot. All attempts of a maze are recorded with the account_id, maze_id, and movement list.

Document Revision History

- Rev. 1.0 2023-10-01: Initial version.
 - Added details to User Requirements, Use Cases, Security Requirements, and System Requirements.
- Rev. 1.1 2023-10-08: Added entity relationship diagram and database schema details to Specification.
- Rev. 1.2 2023-10-10: Updated entity relationship diagram and database schema in Specification.
- Rev. 1.3 2023-11-18: Added details to Interface Requirements.
 - Updated System Requirements.
 - Updated Security Requirements.
- Rev. 1.4 2023-11-20: Added new use cases.
- Rev. 2.0 2023-12-11: Updated user requirements.
 - Added new use cases.
 - Updated entity relationship diagram and database schema.
- Rev. 2.1 2023-12-14: Added user action diagrams for login page, user home page, admin home page, create/edit maze page, and attempt maze page.

Customer

Persons just beginning to learn how to program in Java are this project's primary stakeholders. They will be able to gain, practice and test their knowledge of the basics of Java programming without having to implement the more advanced aspects of programming that normally would be required to simulate the robot and facilitate navigation through the maze environment.

User Requirements

1. Login/create account
 - a. Choose user type (user/admin)
 - b. Choose a password and unique username
 - c. Log in to app upon successful account creation
2. Create/edit maze as admin user
 - a. Select "Create New Maze"
 - b. Click empty grid spaces to create walls, a start, and a finish position
 - c. Click save to create/update the current maze
3. Attempt maze challenge
 - a. Choose a maze to attempt from the catalog (homepage)
 - b. Write code and hit compile to run the code and get a score
 - c. Click save to store the current attempt in the database for future use
4. View past attempts
 - a. Choose maze to view
 - b. Toggle between your best attempt (least moves) or most recent attempt

Use Cases

Name	Create User
Actors	Any user or admin
Triggers	Sign Up button (click)
Events	User chooses a user type (user or admin user), enters a unique username and a password.
Exit Conditions	Required information is entered correctly
Post-conditions	The user is redirected to a homepage.
Acceptance Test	The user can now login with their chosen username and password.

Name	Login User
Actors	Any user or admin
Triggers	Login button (click)
Events	User enters a username and password
Exit Conditions	Information matches an account in the database
Post-conditions	The user is redirected to the homepage of their account type
Acceptance Test	The user now has access to their homepage.

Name	Create maze
Actors	Admin
Triggers	Clicking the New Maze button
Events	The admin user clicks on empty grid spaces to create walls, start, and finish position.
Exit Conditions	Maze has a start and finish position.
Post-conditions	Success snackbar displayed to the admin.
Acceptance Test	The new maze appears on the homepage.

Name	Select a maze
Actors	User
Triggers	Maze Display (click)
Events	The user selects the maze they wish to attempt by clicking on the maze's layout from the catalog.
Exit Conditions	The clicked maze has an id and layout array.
Post-conditions	The user is redirected to the maze attempt page with the maze, robot, and code textfield available.
Acceptance Test	The correct maze is displayed on the screen.

Name	Compile user-input code for completion attempt.
Actors	User
Triggers	Clicking on the compile button
Events	After having written their Java code, the user submits that code for a maze compilation by clicking on the compile button.
Exit Conditions	The code compiles and generates a movement list.
Post-conditions	The movement list enables the Play button for animating the result.
Acceptance Test	The animation of the movement list matches the code submitted by the user.

Name	Save user-input code for future attempts.
Actors	User
Triggers	Clicking the Save button
Events	After having written their Java code, the user saves that code by clicking Save.
Exit Conditions	The code compiles and generates a movement list that is then saved to the database.
Post-conditions	A success snackbar is displayed for 5 seconds.
Acceptance Test	The code under “Most Recent Attempt” matches the submitted code.

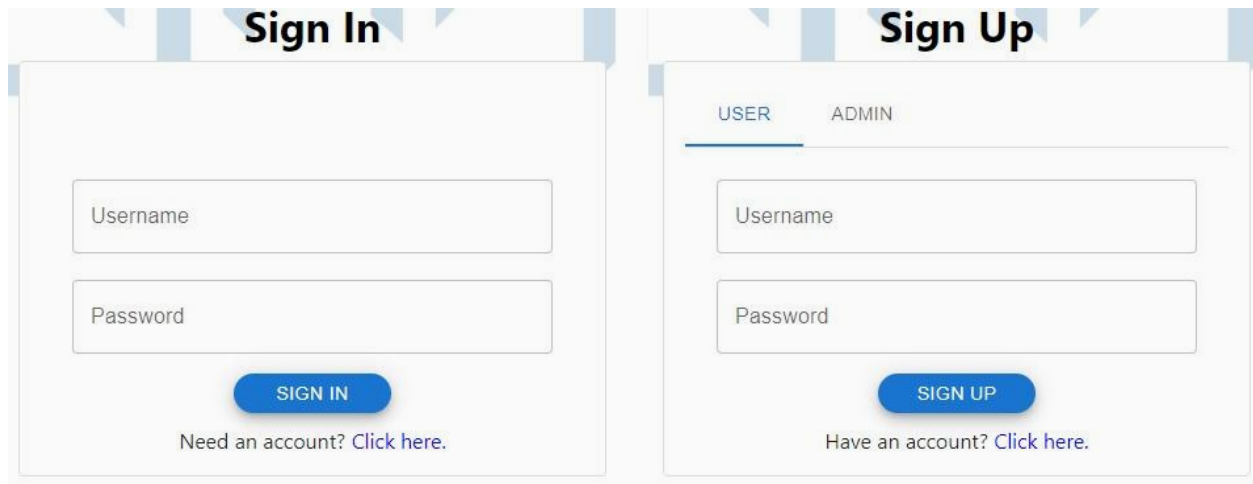
Name	Animate the robot.
Actors	User
Triggers	Clicking the Play button
Events	The currently viewed code is “ran” and the robot moves through the maze.
Exit Conditions	The movement list reaches “Success” or “Failure”
Post-conditions	The score of this attempt is displayed to the user.
Acceptance Test	The displayed score matches the number of times the robot moved.

User Interface Requirements

The website is divided into two main views: the user view and the admin view. The user view is designed to allow users to write java code that will navigate the robot through a chosen maze, while the admin view is designed to allow admins to create or delete mazes.

Login Page

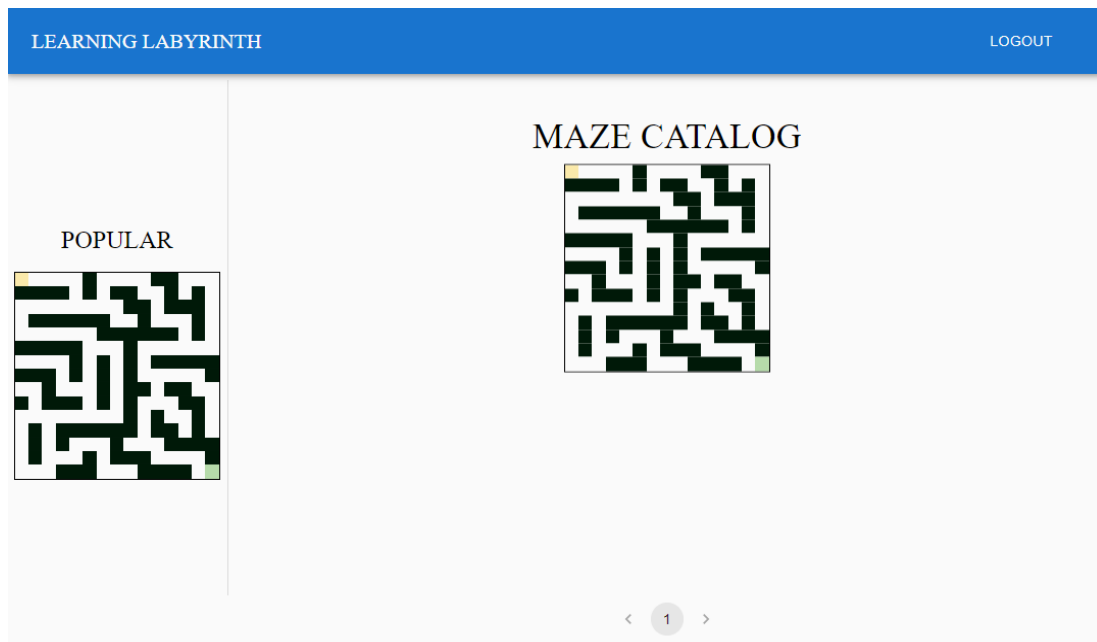
The login page has two configurations: the Sign In page for users and admins with accounts, and the Sign Up page for those who need to create an account.



The image shows two side-by-side login forms. The left form is titled 'Sign In' and contains fields for 'Username' and 'Password', a blue 'SIGN IN' button, and a link 'Need an account? Click here.' The right form is titled 'Sign Up' and has tabs for 'USER' and 'ADMIN'. It contains fields for 'Username' and 'Password', a blue 'SIGN UP' button, and a link 'Have an account? Click here.'

User View

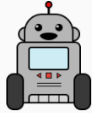
The user view is divided into two pages: the user home page and the attempt maze page. The user home page allows the user to select a maze to attempt.



The attempt maze page allows users to write Java code that will attempt to navigate the robot through the maze. Hovering the mouse over the robot's image displays an information tooltip showing the methods that can be called to navigate the robot.

LEARNING LABYRINTH

LOGOUT



New code attempt initiated! Keep working to develop a new method. Remember to hover over me to see my available methods!

BEST CODE

MOST RECENT CODE

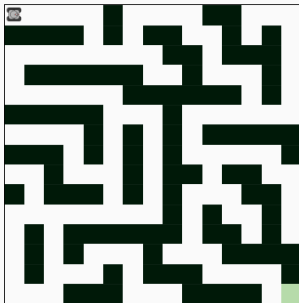
CURRENT CODE

Your Code Here

```
do {
  robot.moveForward();
} while (robot.scanForward() == GridTypeEnum.PATH);
```

COMPILER

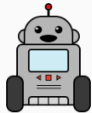
SAVE



▶ PLAY

LEARNING LABYRINTH

LOGOUT



New code attempt initiated! Keep working to develop a new method. Remember to hover over me to see my available methods!

BEST CODE

MOST RECENT CODE

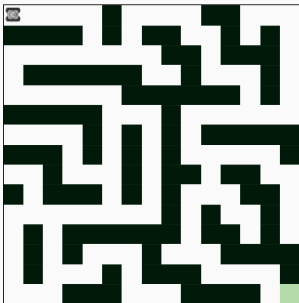
CURRENT CODE

Your Code Here

```
do {
  robot.moveForward();
} while (robot.scanForward() == GridTypeEnum.PATH);
```

COMPILER

SAVE



▶ PLAY

Enums

New code attempt initiated! Keep working to develop a new method. Remember to hover over me to see my available methods!

GridTypeEnum (WALL, PATH, FINISH)

Scan Methods

robot.scanForward(): returns GridTypeEnum of space in front of M.A.Z.E.R.

robot.scanBackward(): returns GridTypeEnum of space behind M.A.Z.E.R.

robot.scanRight(): returns GridTypeEnum of space to right of M.A.Z.E.R.

robot.scanLeft(): returns GridTypeEnum of space to left of M.A.Z.E.R.

Move Methods

robot.moveForward(): moves M.A.Z.E.R. forward 1 space

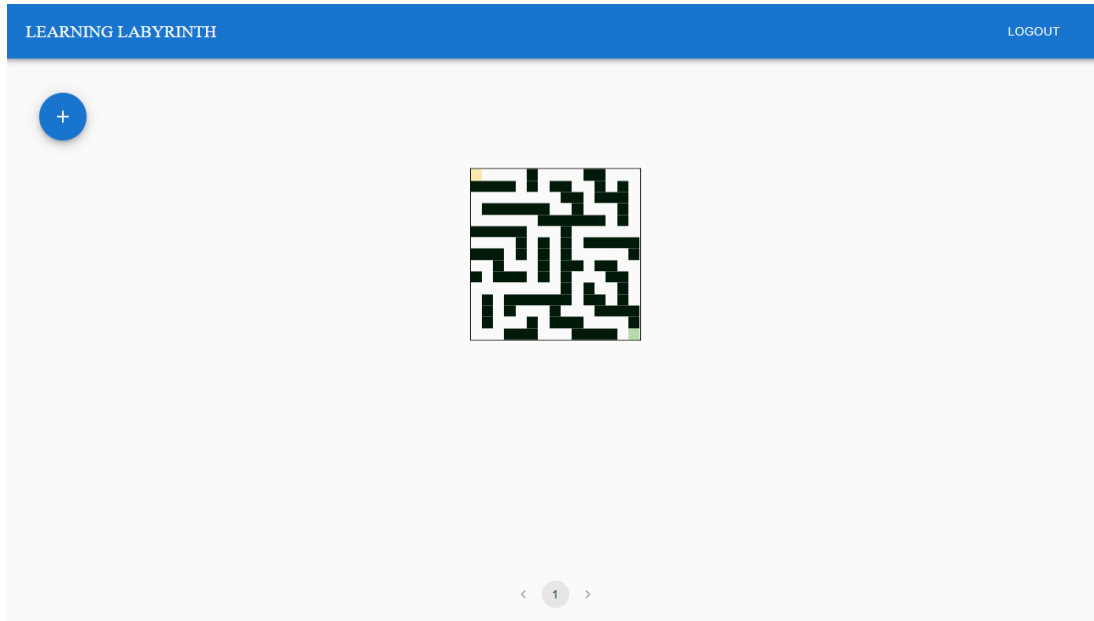
robot.moveBackward(): moves M.A.Z.E.R. backward 1 space

robot.rotateRight(): rotates M.A.Z.E.R. right 90 degrees

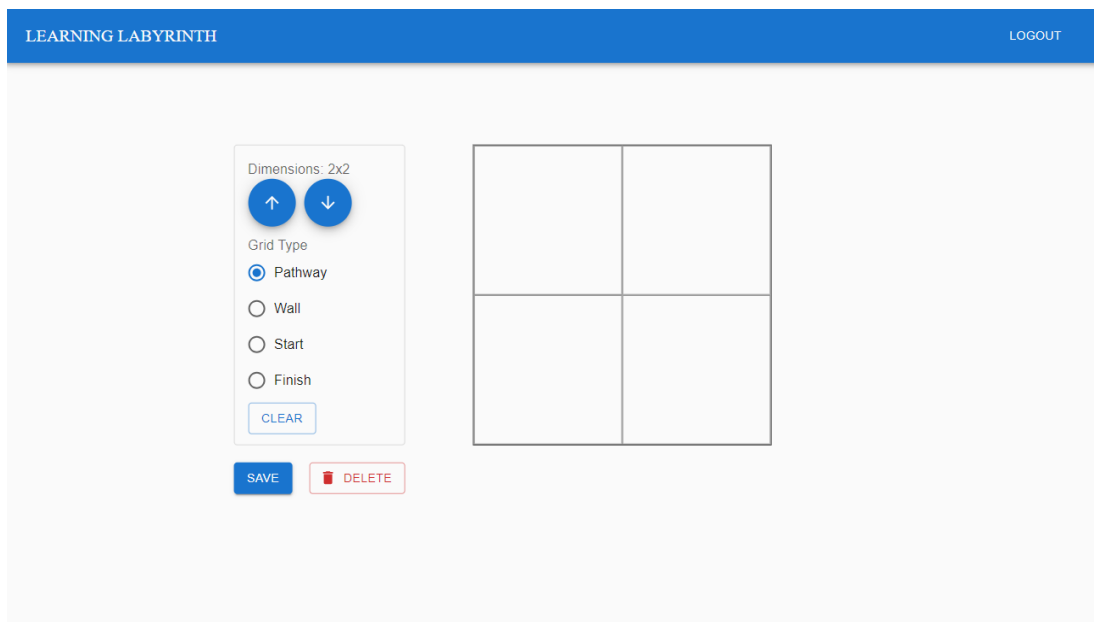
robot.rotateLeft(): rotates M.A.Z.E.R. left 90 degrees

Admin View

The admin view is divided into two pages: the admin home page and the create maze page. The admin home page allows an admin to choose any of the existing mazes for editing or deletion, or begin the creation of a new maze.



The create/edit maze page is where an admin can either edit or delete an existing maze, or create a new maze. All mazes must have a start location and finish location in order to be saved to the database.



Security Requirements

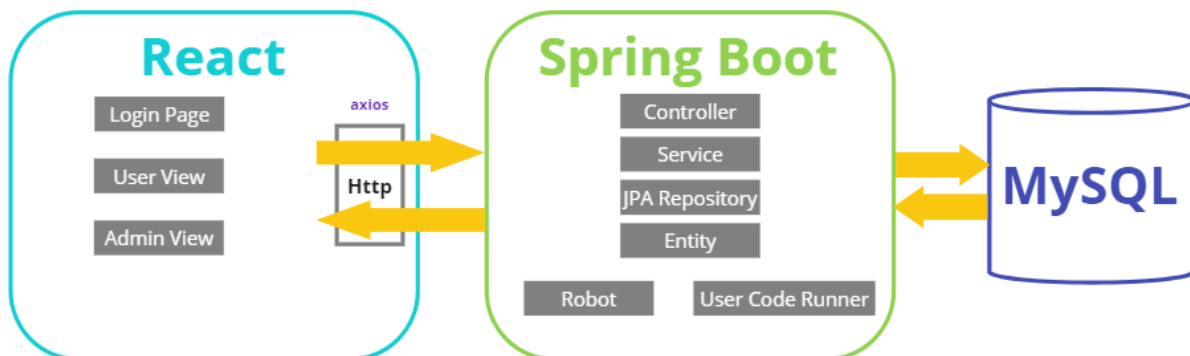
- Two distinct roles with different permissions: User and Admin
- User authentication necessary
- No user should be able to modify the data of another user, unless as an admin.
- Account creation is required to use this application
 - A hashmap at runtime stores tokens of all logged in users consisting of a randomly generated UUID as key and an Account object as value
 - Tokens are created in the hashmap upon user login, and removed from the hashmap upon user logout
- Password hashes are stored in the database rather than in plaintext form
- User code should be run in a new process to isolate any malicious or unintended behavior resulting from said code

System Requirements

- Server Requirements
 - Backend: Java Spring Boot
 - ORM: JPA Repository
 - Build Tool: Gradle
 - Frontend: Javascript React
 - Runtime Environment: Node.js
 - Database: MySQL
- User Requirements
 - Modern Web Browser(Desktop)
 - Stable Internet Connection

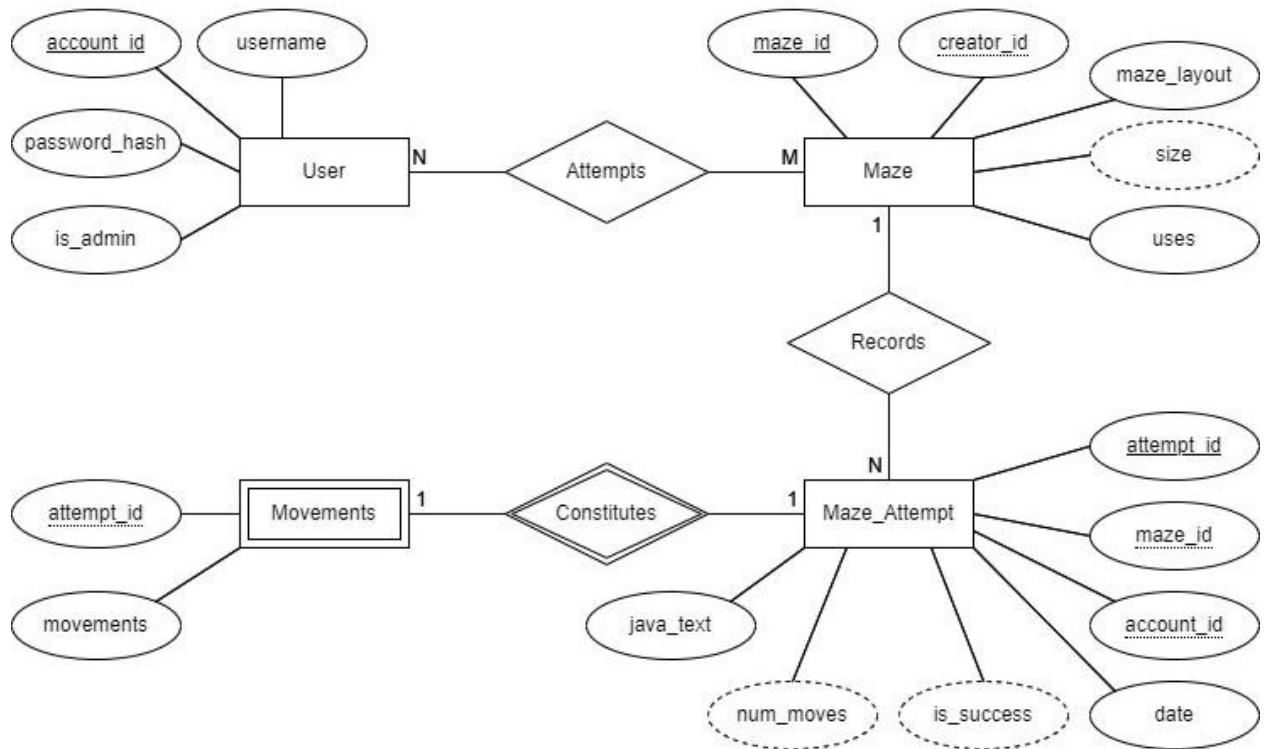
Specification

Application Architecture:



Database Architecture:

Entity Relationship Diagram (Chen Notation):



Database Schema:

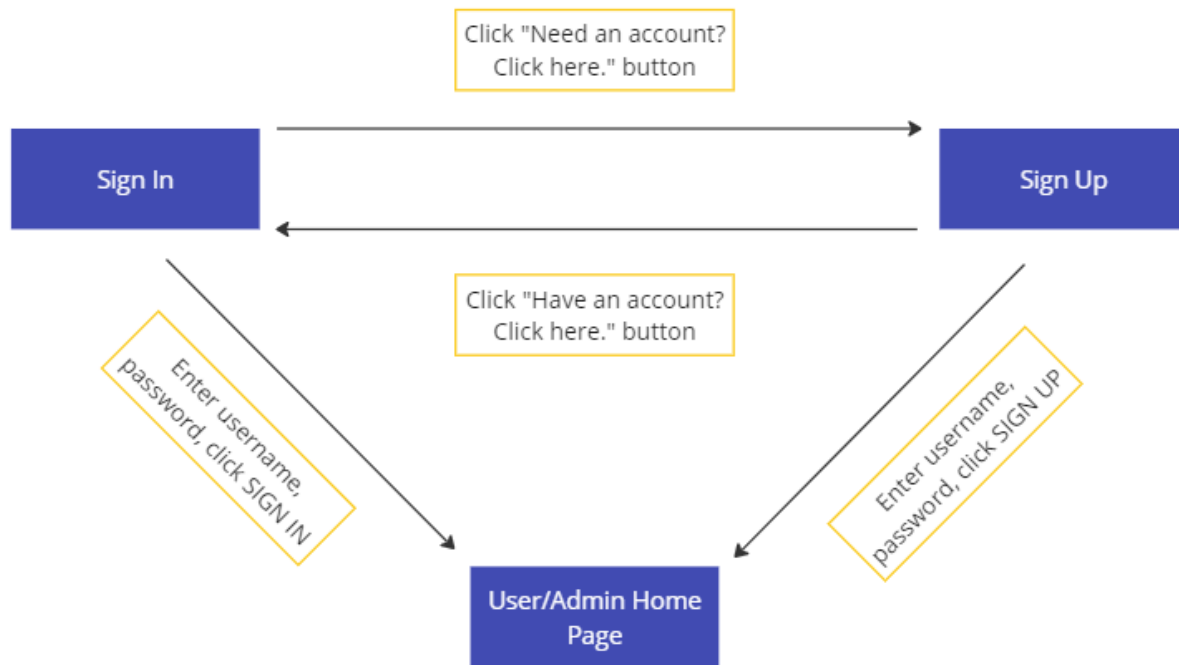
user(id(PK), username, password_hash, is_admin)

maze(id(PK), layout, creator_id, size, uses)

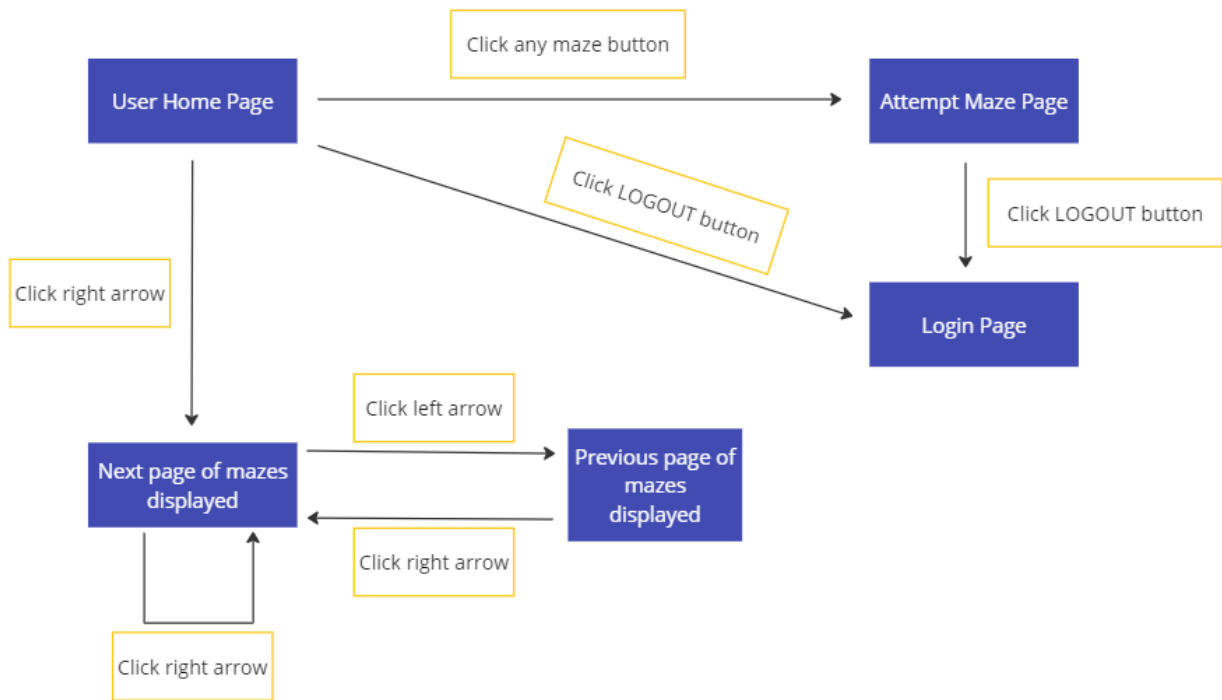
maze_attempts(id(PK),
 maze_id,
 account_id,
 date,
 is_success,
 num_moves,
 java_text,
 movements(FK))

movements(attempt_id, movements)

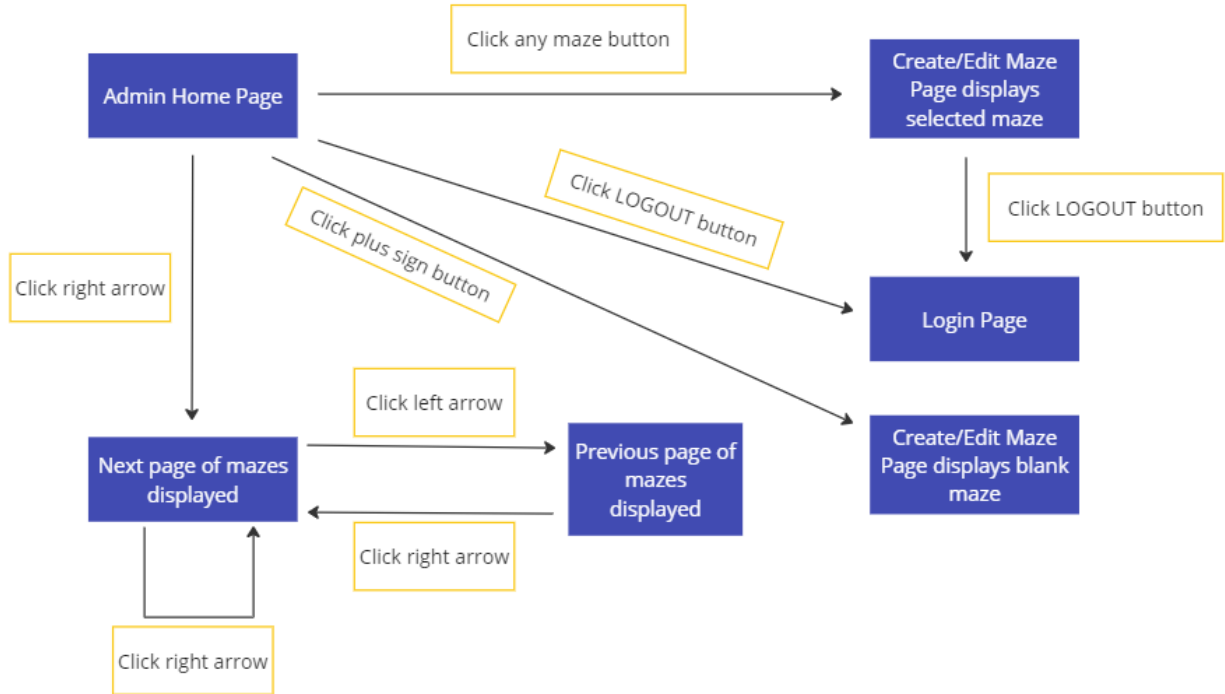
User Action Diagram for Login Page:



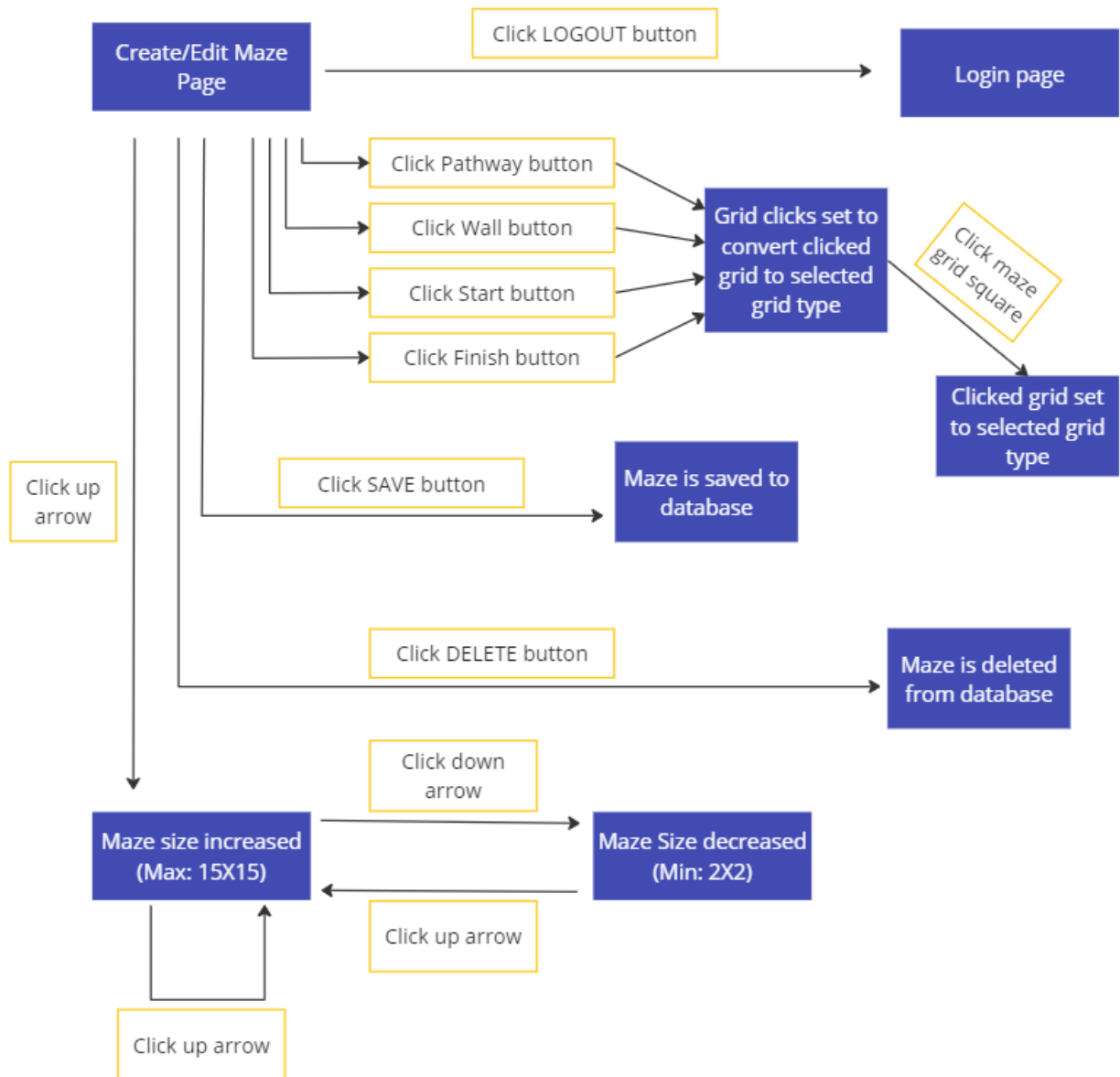
User Action Diagram for User Home Page:



User Action Diagram for Admin Home Page:



User Action Diagram for Create/Edit Maze Page:



User Action Diagram for Attempt Maze Page:

