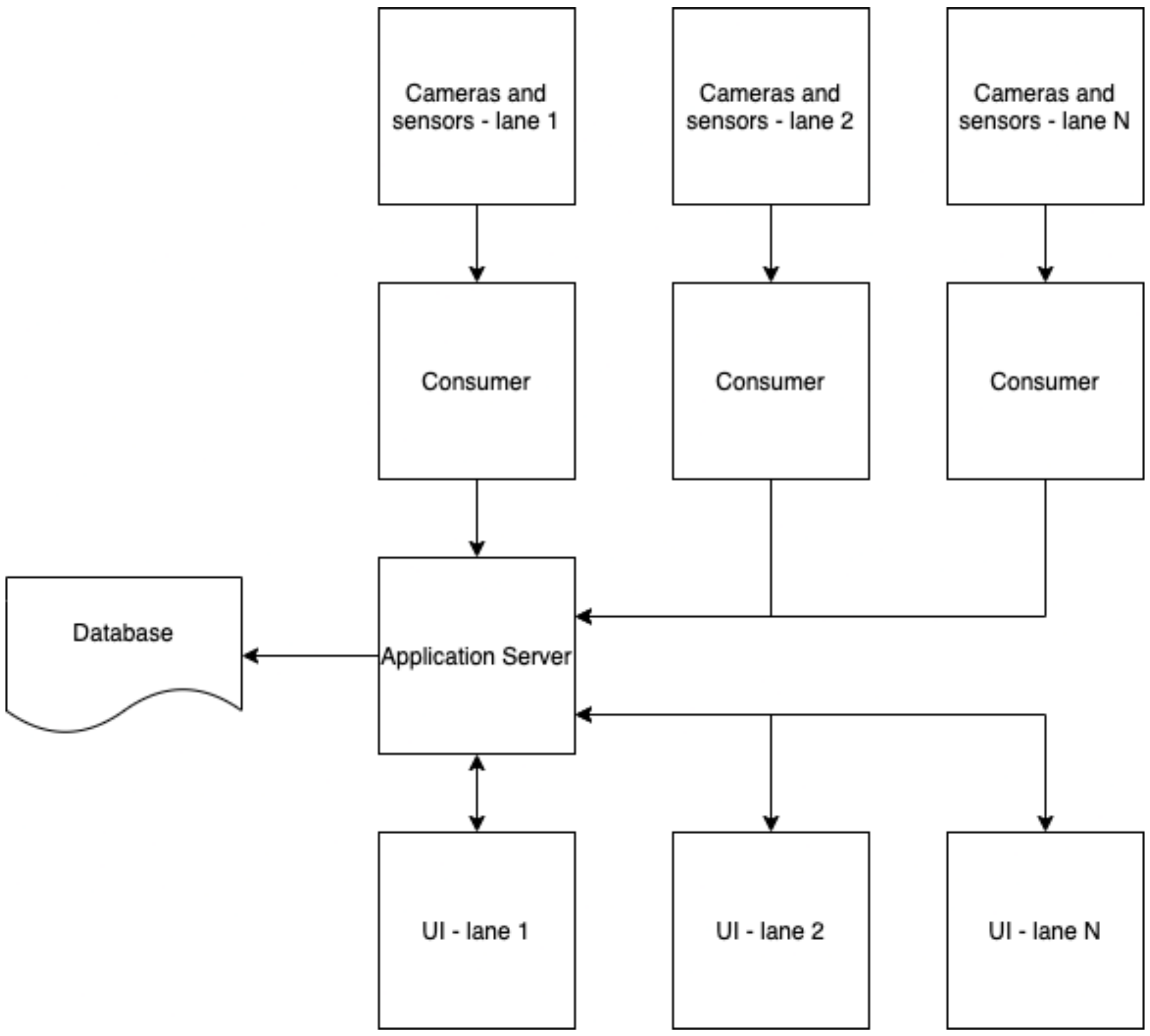
Candlepin Automated Scoring

Problem Statement

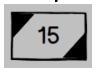
A candlepin bowling company wants to upgrade their bowling lanes with an automated scoring system. The lanes have been outfitted with cameras and sensors which will feed information about downed pins to an application server. This server should calculate and track game information for each lane. Each lane will have a tablet that displays the game information for all players to see.

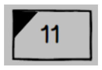
Architecture Diagram

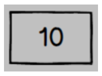

Cameras and sensors - lane 1
Cameras and sensors - lane 2
Cameras and sensors - lane N
Consumer
Consumer
Consumer
Database
Application Server
UI - lane 1
UI - lane 2
UI - lane N

UI Requirements

Below are the application requirements from the frontend perspective. This should be an application that:

1. Runs on a tablet (screen resolution is 1280 x 800)
    - Extra credit: runs on mobile as well
2. Displays the lane number and the elapsed time
    - Time begins when an admin (e.g. the manager at the bowling alley) starts the session
3. Allows users to add at least one user, but no more than 8 users to a game
    - See wireframes “Main” and “Start”
4. Displays the current bowler and the frame that is currently being played. This user’s name and the frame to bowl should both be highlighted.
    - The left column is the number of downed pins in that frame
    - The right column is the running total of all pins downed
        - **Except** for the first frame. In the first frame, the left box is empty and only the right box is filled with the total number of downed pins
    - **Note:** These calculations are handled by the backend, the frontend will only be *displaying* the values – no calculations necessary
    - There are special symbols in the game:
        - 

 - two triangles denote a STRIKE. The number in the center is 10 plus the number of pins downed in the next TWO rolls (not frames). This information will be provided by the API.
        - 

 - one triangle denotes a SPARE. The number in the center is 10 plus the number of pins downed in the NEXT roll (not frames). This information will be provided by the API.
        - 

 - in candlepin, 10 pins can be downed without a mark since the roller gets THREE rolls per frame. A 10 simply means the roller used all THREE rolls to down all pins.
    - See wireframe “Frame”
5. Provides a button to restart the game. This button is only visible once a game has started. If the game is restarted, the user returns to the state where they must enter users into the game (before game start – see requirement 3).
    - See wireframe “Restart”
6. Displays a modal when the game ends (e.g. all players have rolled 10 frames). The user is given two options:
    - Restart the game with the current players. This resets the app to requirement 4 with the current players
    - Quit the game. This resets the app to requirement 3 with the players needing to enter users into the game.
    - See wireframe “Game End”

Exercise Requirements

The goal of this exercise to create a *frontend application*. The backend is a “black box” and implementing it will be extra credit. We describe the game logic and information passed between the frontend and backend below to provide a better understanding of this fake system. Some test data is provided in the “test-data” directory to make testing and development easier. Feel free to add your own data as well. Please:

1. Setup:
    a. Create a repository and push the setup code that we’ve provided to two branches: “main” and “devel”
    b. Add a LICENSE file denoting that either: it’s either open source or that you own it. This exercise is for us to review your code, problem solving, and design skills.
    c. Update the README file to contain instructions for building and/or running the application.
    d. Provide read access to the repository to joseph.stone@dataductus.com and scott.barvick@dataductus.com
2. Frontend Application:
    a. Create a new branch off of devel to which the application code will be pushed.
    b. React and Bootstrap are already installed, but feel free to replace with any framework and CSS library of your choice.
    c. You may assume that this will only run in Chrome.
    d. Time permitting, please add unit tests (infrastructure is provided in the setup).
    e. Using the requirements defined above, create the frontend application.
3. Backend, black-box, game logic:
    a. This UI *will not* be performing any game logic. The UI will poll (or wait) for updates from the server about game state. For example: when Rachel downs 5 pins on her first roll, the server will update the game state in the database, and then either notify the client of the update and pass the game state or changes to the frontend OR the client will poll the database for the game state or recent changes. The method by which the UI gets the information, polling or otherwise, should be determined by you.
    b. Important system events:
        i. An admin starts the lane (which starts the elapsed time - server notifies frontend)
        ii. A user adds bowlers to the game
            1. Frontend notifies server, passing the names of the bowlers
        iii. A user starts the game
            1. Frontend notifies server that a game has started. Server will reset the lane and monitor for downed pins.
        iv. A bowler downs pins
            1. Server notifies the frontend with either an object representing the full state of the game OR an object specifying the bowler,

downed pins, whether it’s a strike or spare (this may be null), roll number, and frame

v. A bowler finishes a frame
   1. Server notifies the frontend with either an object representing the full state of the game OR an object specifying bowler, the completed frame, downed pins, and whether it’s a strike or spare (this may be null)

vi. All bowlers have finished 10 frames
   1. Server notifies the frontend that the game is complete. An object containing the full state of the game is provided.

vii. A user restarts or quits the game
   1. Frontend notifies server. No data is needed to be passed.

4. Submitting:
   a. When complete, create a pull request (PR), add us as reviewers, and notify us that the code is ready to be reviewed. We’ll review and make comments or ask questions in the PR.
   b. We will have a final interview in which this application will be used as a discussion point.

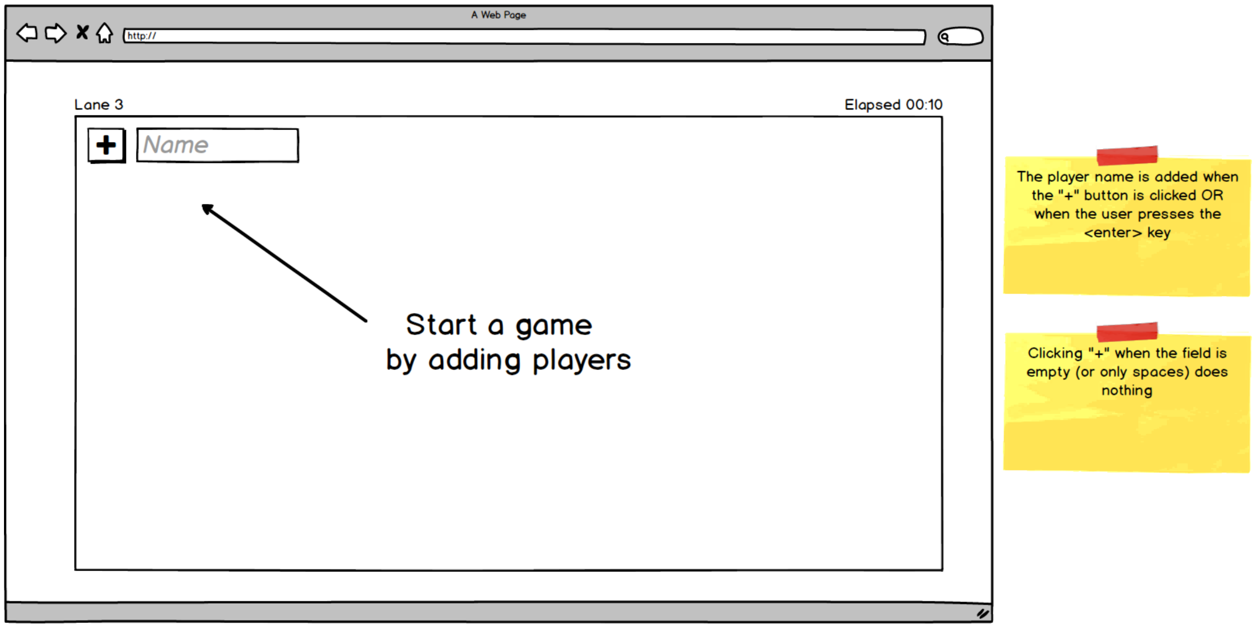
Lastly, have fun! While this exercise is intended for us to review your skills, we want this to be enjoyable. We understand that this application will not be perfect or free of issues. Please see what you can accomplish in 4 hours, but no more than one week.

Please feel free to ask for clarification. This is a new exercise, so there may be missing information or confusing verbiage.

Enjoy!

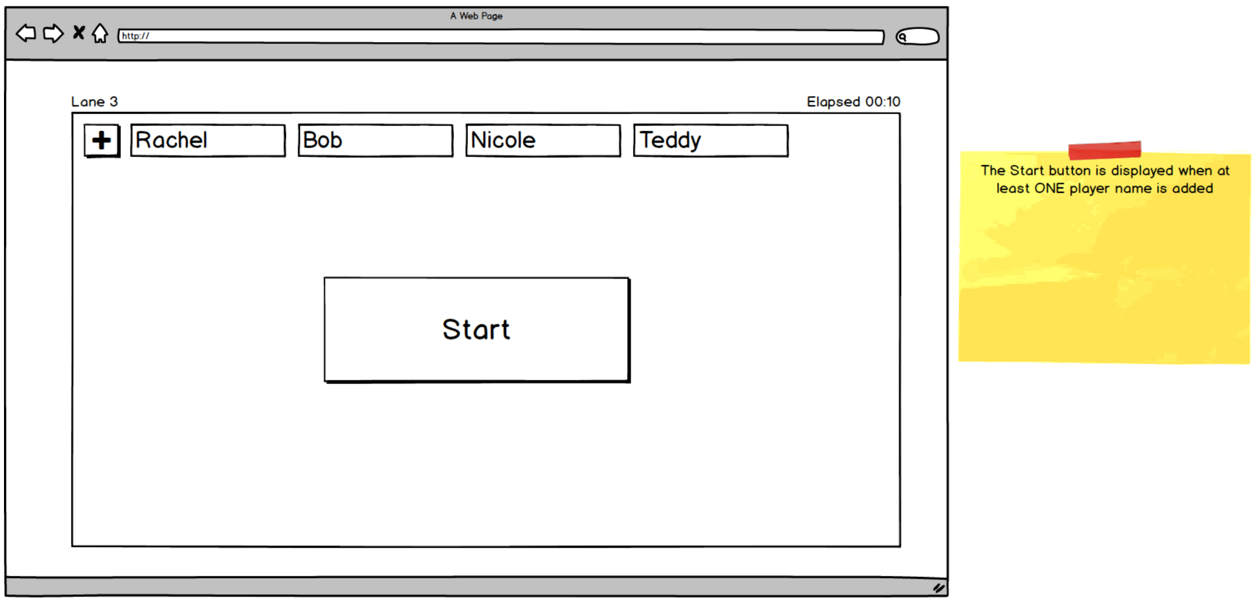
Appendix A – Wireframes

Main



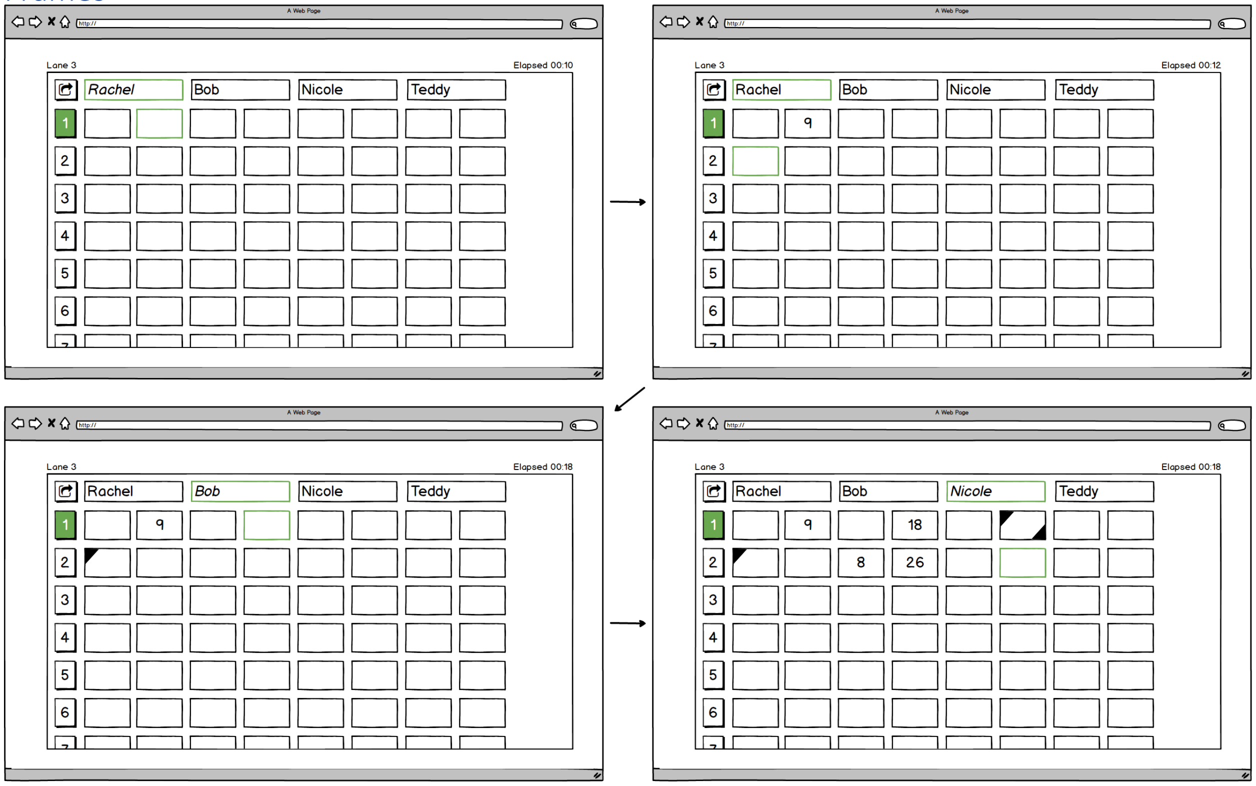
A Web Page
http://
Lane 3
Elapsed 00:10
Name
Start a game by adding players
The player name is added when the "+" button is clicked OR when the user presses the <enter> key
Clicking "+" when the field is empty (or only spaces) does nothing

Start

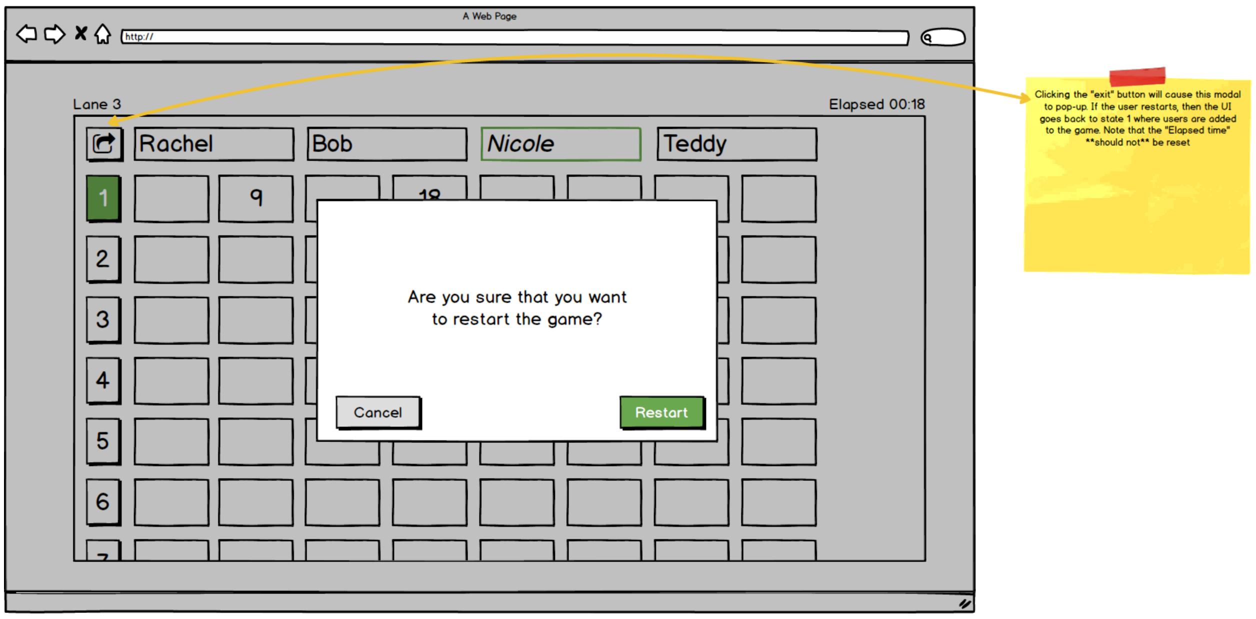


A Web Page
http://
Lane 3
Elapsed 00:10
Rachel
Bob
Nicole
Teddy
Start
The Start button is displayed when at least ONE player name is added

Frames



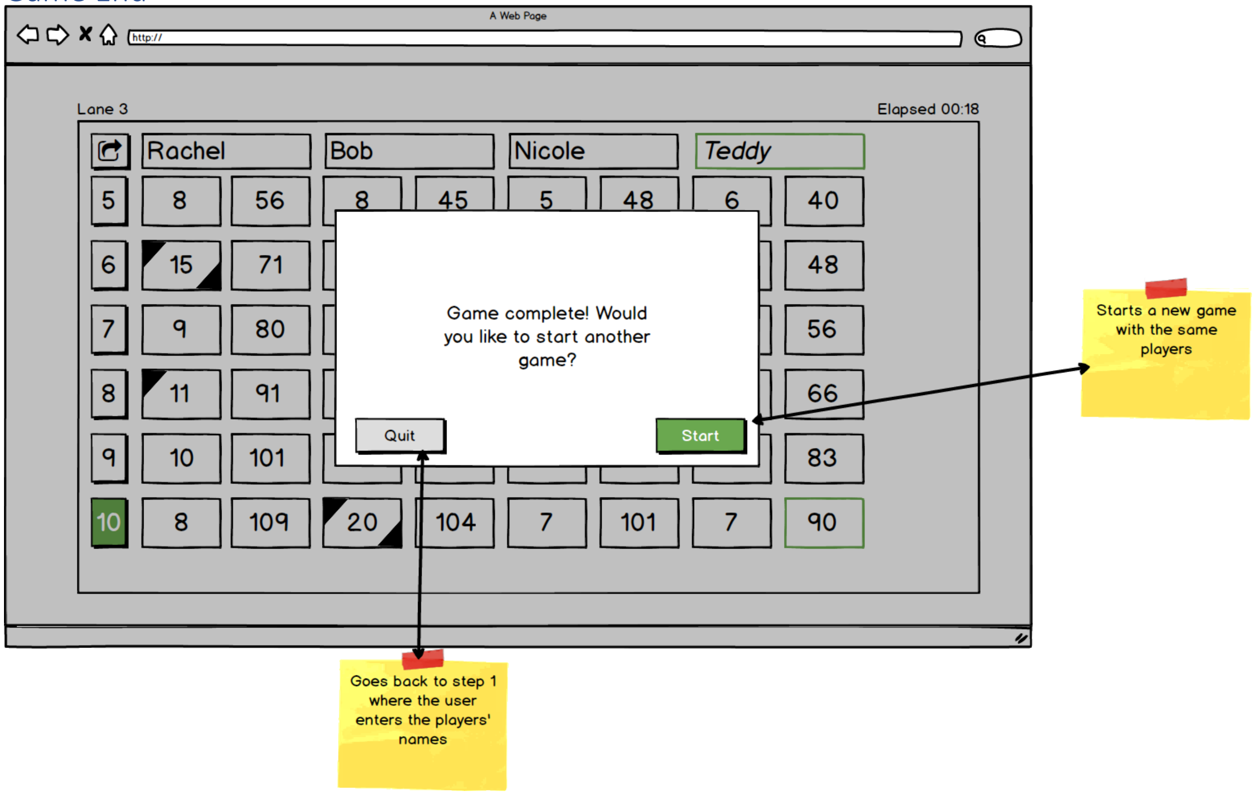
Restart



Are you sure that you want to restart the game?

Clicking the "exit" button will cause this modal to pop-up. If the user restarts, then the UI goes back to state 1 where users are added to the game. Note that the "Elapsed time" **should not** be reset

Game End



Game complete! Would you like to start another game?

A Web Page
http://
Lane 3
Elapsed 00:18
Rachel
Bob
Nicole
Teddy
Quit
Start
Starts a new game with the same players
Goes back to step 1 where the user enters the players' names