NAVIGATE THE MARS ROVER

Problem statement: To find the shortest path between two given points by avoiding all the obstacles between the end points.

Project Scope

Project Title: **NAVIGATE THE MARS ROVER** Date **: 07/06/2020** Prepared By **: The Aatmnirbhars**
Project Justification: This project is basically making system for the Mars crew pin accurate because no line of error is required.
Product Characteristics and Requirements : 1. Use of best Algorithms. 2. Tracking of rover. 3. Search features. 4. Calculating distance covered if any changes occur due to the path change.
Project Management Deliverables: Project plan, Deployment of project, Report and review, Software codes, Design documents
Project Success Criteria: We aim to complete this project within time with the use of best algorithms available and improve the latest technology known by us.

Statement of Work

1. Scope of Work:

Welcome to Pathfinding Visualizer! We built this application because we were fascinated by pathfinding algorithms and we wanted to visualize them in action. We hope that you enjoy playing around with this visualization tool just as much as we enjoyed building it.

2. Location of Work

The user for this project is the rover going to land on mars first time in human history.

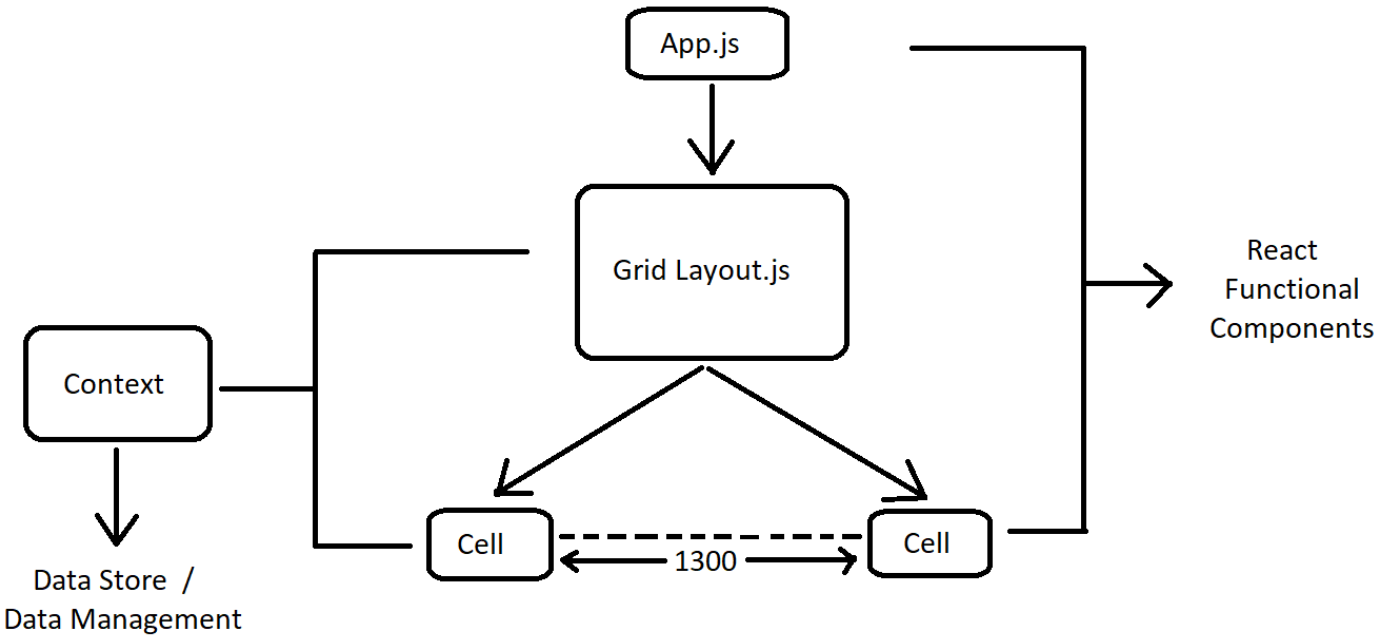
3. Period of Performance

The duration of the project is 1 month.

Deliverables Schedule

Deliverable	Date Due
Statement of work and Project Plan	Week1
Project scope & problem statement/ requirements specification	Week 2
Problem analysis	Week 3
Prototype implementation or practical experimental work	Week 4
Review of results and statement of findings (including test reports)	Week 4
Status reports and project diary	Completed and on review with mentors

Getting Started



Create React App is an officially supported way to create single-page React applications. It offers a modern build setup with no configuration.

#Quick Start

```
npx create-react-app my-app cd
my-app
```

```
npm start
```

If you've previously installed `create-react-app` globally via `npm install -g createreact-app`, we recommend you uninstall the package using `npm uninstall -g create-react-app` to ensure that `npx` always uses the latest version.

Then open http://localhost:3000/ to see your app.

When you're ready to deploy to production, create a minified bundle with `npm run build`.

Get Started Immediately

You don't need to install or configure tools like webpack or Babel. They are preconfigured and hidden so that you can focus on the code.

Create a project, and you're good to go. s

Creating an App

You'll need to have Node >= 8.10 on your local development machine (but it's not required on the server). You can use nvm (macOS/Linux) or nvmwindows to switch Node versions between different projects.

To create a new app, you may choose one of the following methods:

#npx

```
npx create-react-app my-app
```

#npm

```
npm init react-app my-app
```

`npm init <initializer>` is available in npm 6+

#Selecting a template

You can now optionally start a new app from a template by appending `_template [template-name]` to the creation command.

If you don't select a template, we'll create your project with our base template.

Templates are always named in the format `cra-template-[template-name]`, however you only need to provide the `[template-name]` to the creation command.

```
npx create-react-app my-app --template [template-name]
```

You can find a list of available templates by searching for "cra-template-*" on npm.

Our Custom Templates documentation describes how you can build your own template.

#Creating a Typescript app

You can start a new TypeScript app using templates. To use our provided TypeScript template, append `--template typescript` to the creation command.

```
npx create-react-app my-app --template typescript
```

If you already have a project and would like to add TypeScript, see our Adding Typescript documentation.

#Selecting a package manager

When you create a new app, the CLI will use Yarn to install dependencies (when available). If you have Yarn installed, but would prefer to use npm, you can append `--use-npm` to the creation command. For example:

```
npx create-react-app my-app --use-npm
```

#Output

Running any of these commands will create a directory called `my-app` inside the current folder. Inside that directory, it will generate the initial project structure and install the transitive dependencies:

```
my-app ├──
README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   ├── logo192.png
│   ├── logo512.png
│   ├── manifest.json
│   └── robots.txt
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js
```

No configuration or complicated folder structures, only the files you need to build your app. Once the installation is done, you can open your project folder:

```
cd my-app
```

#Scripts

Inside the newly created project, you can run some built-in commands:

#`npm start`

Runs the app in development mode. Open http://localhost:3000 to view it in the browser.

The page will automatically reload if you make changes to the code. You will see the build errors and lint warnings in the console.

#npm test

Runs the test watcher in an interactive mode. By default, runs tests related to files changed since the last commit.

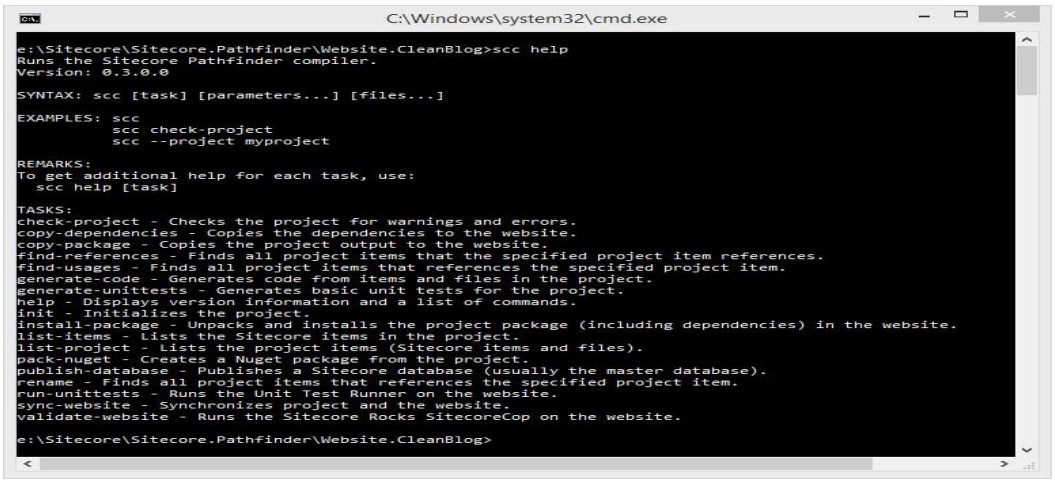
Read more about testing.

#npm run build or yarn build

Builds the app for production to the `build` folder. It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.

Your app is ready to be deployed.



Meet the Algorithms of Project

This application supports the following algorithms:

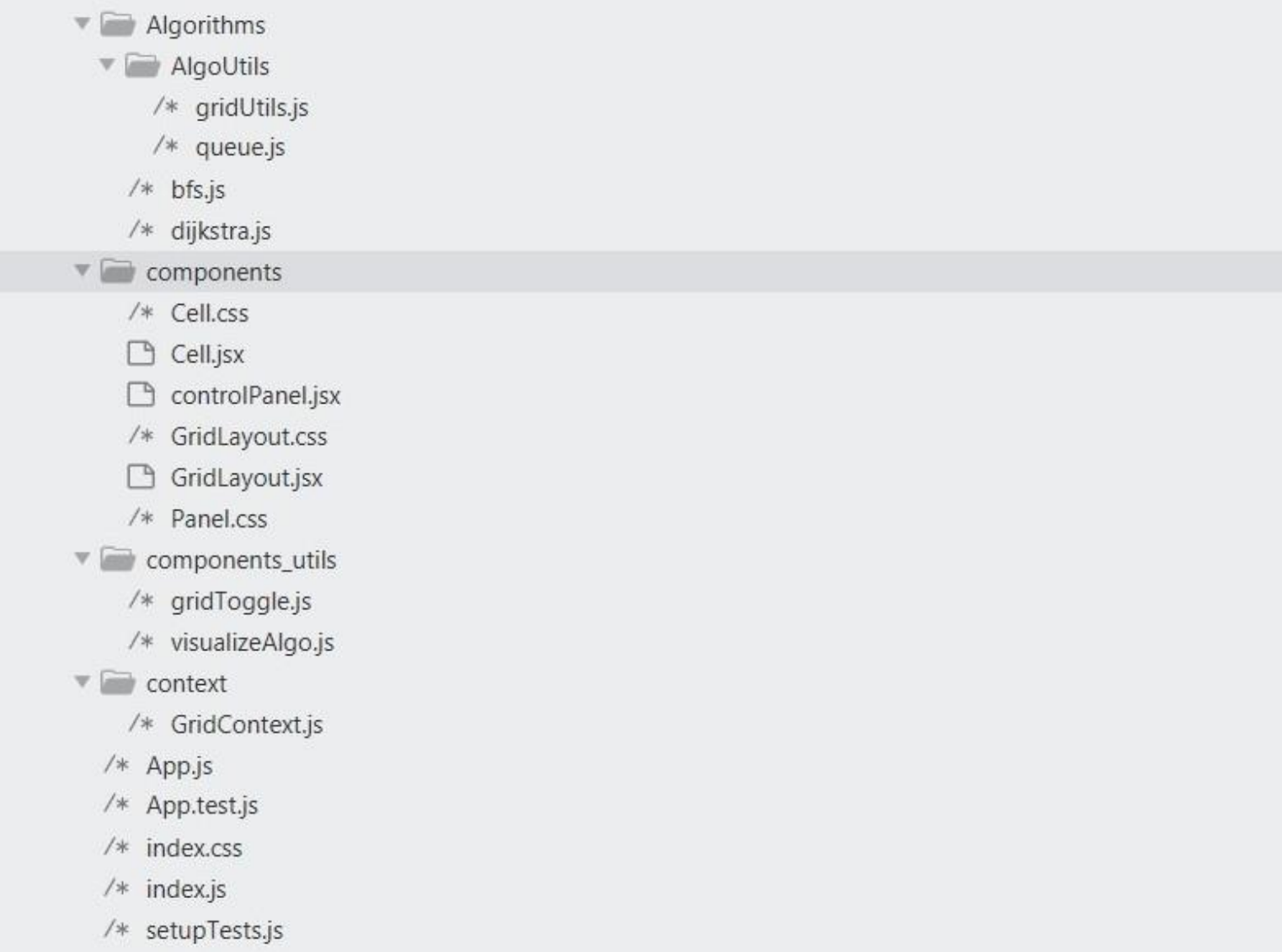
Dijkstra's Algorithm (weighted): the father of pathfinding algorithms; guarantees the shortest path

Breath-first Search (unweighted): a great algorithm; guarantees the shortest path

Depth-first Search (unweighted): a very bad algorithm for pathfinding; does not guarantee the shortest path

On top of the pathfinding algorithms listed above, we implemented a **Recursive Division** Maze Generation algorithm.

Project Structure



Folder Description -

1) **Algorithm Folder :** We have our two Algorithms here. And AlgoUtils file consists of common functions that were used by the algorithm. Like QUEUE, FindNeighbours.

2) **components Folder :** Component file component contains 3 components

1) GridLayout : It is a wrapper for 1300 cells.
   All cells are child component of this component.
2) Cell: It is a single unit of our GridLayout.

3) ControlPanel: It contains all the Algorithm controller like clear walls, clear path.

3) components_utils Folder : It contains supporting modules for our components like:

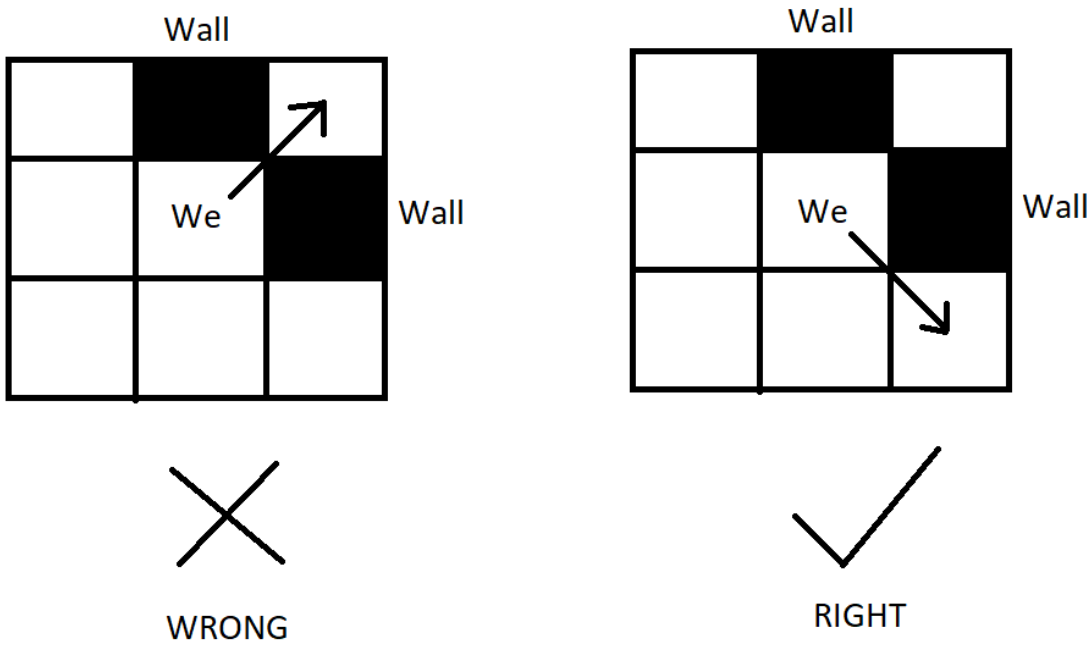
1) gridToggle.js : It's role is to control creating walls, resetting walls, initialising grid(all cells) and reset the path.
2) visualizeAlgo : its role is to visualise the algorithm simulation in DOM, i.e. UI Front End.

4) context Folder : It is a state manager/data store of our application. It manages states of all the cells - every single one of the 1300 cells. All the mouse handler is also here, that is used to toggle state of cells.

More about the Dijkstra's algorithm

Dijkstra's algorithm (or Dijkstra's Shortest Path First algorithm, SPF algorithm) is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road network. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and was published three years later. The algorithm exists in many variants. Dijkstra's original algorithm found the shortest path between two given nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree. For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road (for simplicity, ignore red lights, stop signs, toll roads and other obstructions), Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. A widely used application of shortest path algorithm is network routing protocols, most notably IS-IS (Intermediate System to Intermediate System) and Open Shortest Path First (OSPF).

DIAGONAL APPROACH



More about the BFS search

**Breadth-first search (BFS) algorithm **

Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root or some arbitrary node of a graph, sometimes referred to as a 'search key', and explores all of the neighbour nodes at the present depth prior to moving on to the nodes at the next depth level. Input: A graph G and a starting vertex root of G.

Output: Goal state. The parent links trace the shortest path back to root

1 procedure BFS(G, root) is 2 let Q be a queue 3 label root as discovered 4 Q.enqueue(root) 5 while Q is not empty do 6 v := Q.dequeue() 7 if v is the goal then 8 return v 9 for all edges from v to w in G.adjacentEdges(v) do 10 if w is not labeled as discovered then 11 label w as discovered 12 parent := v 13 Q.enqueue(w)

Functional Requirements

- Operating System: Windows 9x or above, MAC or UNIX.
- Processor: Pentium III or 2.0 GHz or higher.
- RAM: 256 Mb or more

Software Interfaces

- **Application**: React ,Node Js
- **Web Server: Any browser with enabled java script Communications Interfaces:** The Customer must connect to the Internet to access the app

Other Non-functional Requirements

Performance Requirements

The proposed system that we are going to develop will be used in any system within the different browser. Therefore, it is expected that the project would perform functionally all the requirements that are specified by the problem statement.

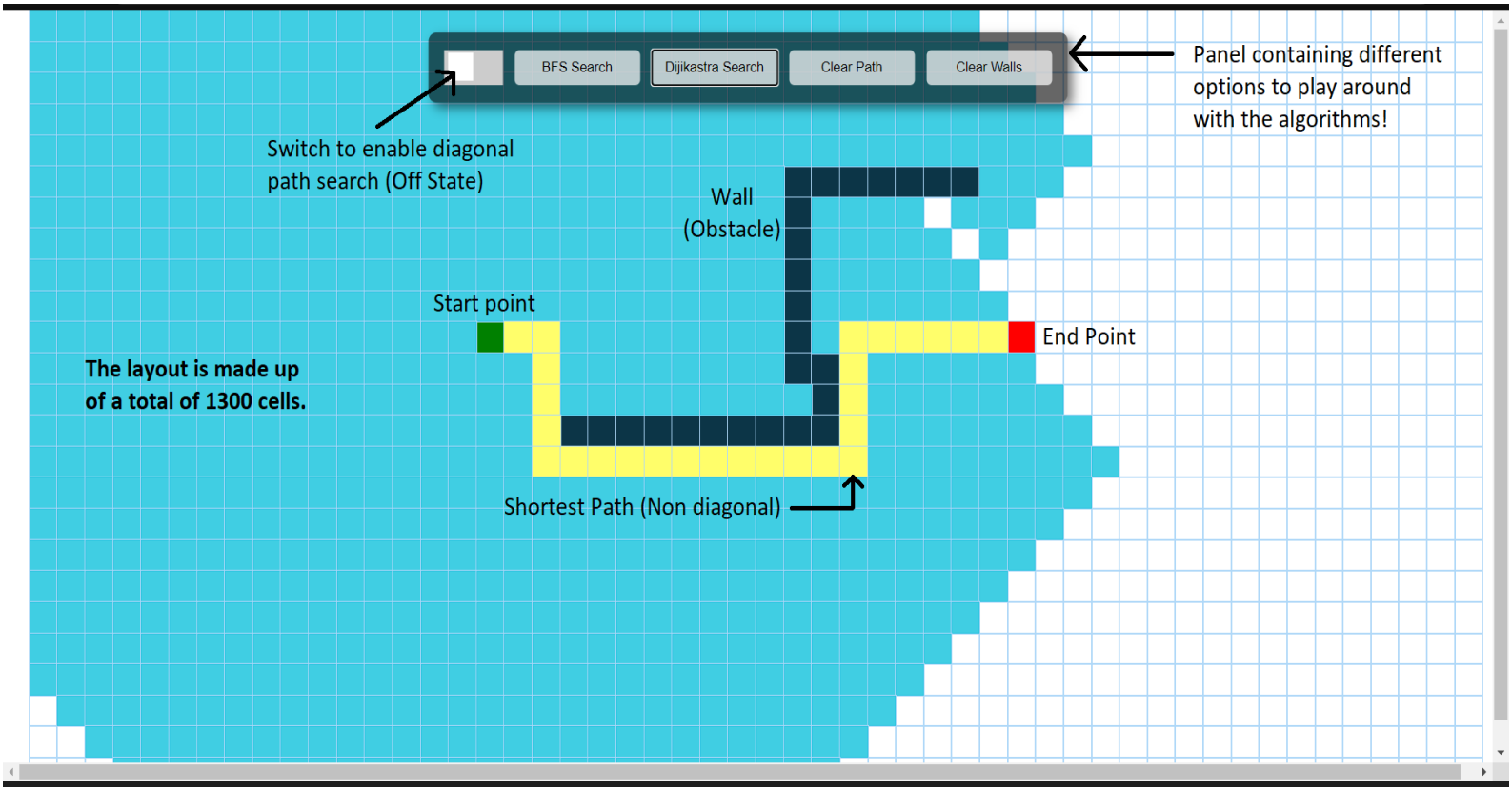
Safety Requirements - The database may get crashed at any certain time due to invalid inputs .

Software Quality Attributes - The Quality of the software is maintained in such a way so that it can be very user friendly to all the users of the database.

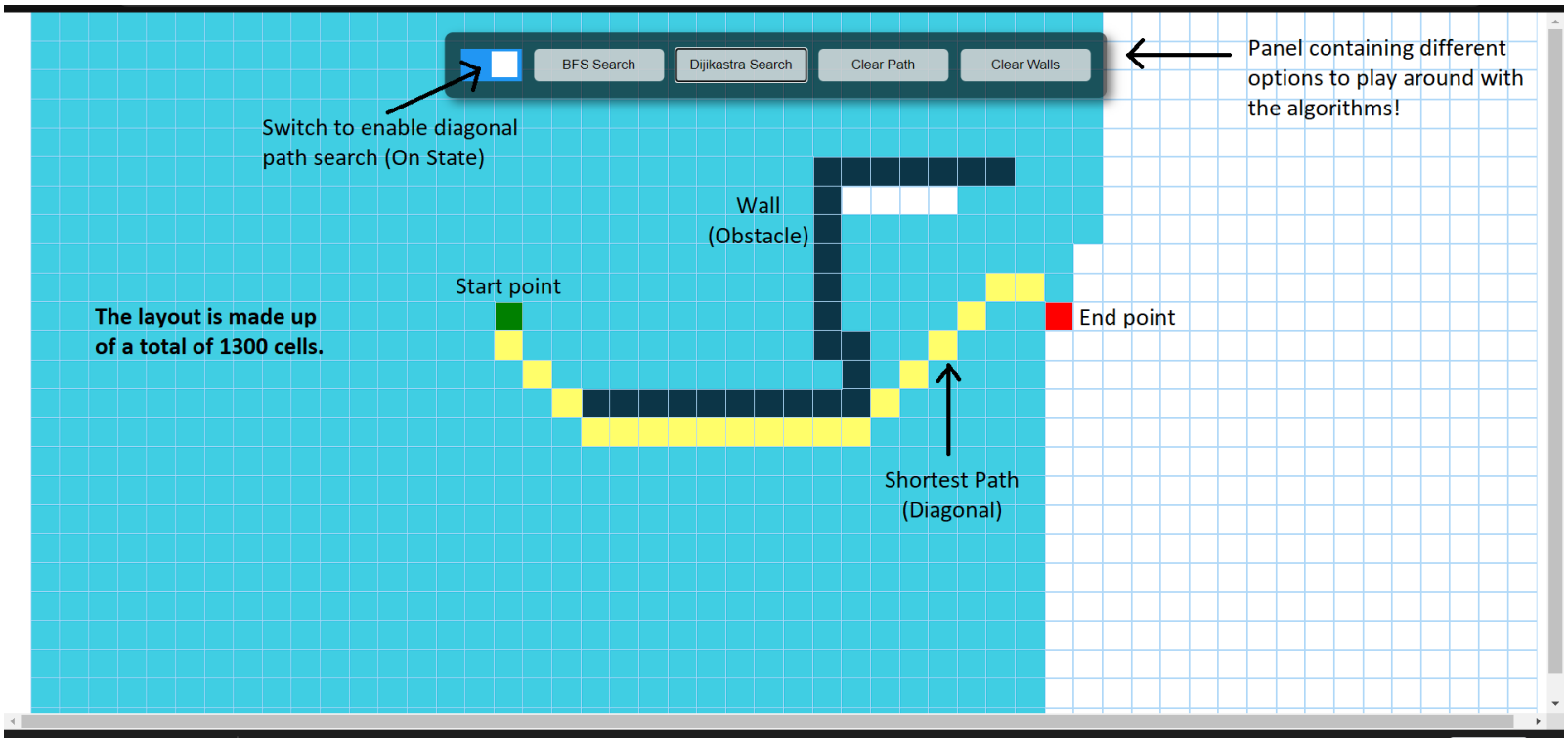
Software Constraints - The development of the system will be constrained by the availability of required software such as Node Js, React and permission of JavaScript. The most recent versions of software development tools may not be installed .

Design Constraints - The system must be designed to allow web usability. That is, the system must be designed in such a way that will be easy to use and visible on most of the browsers.

Our hardwork looks exactly like the following!



1) Shortest path depiction : Non Diagonal



2) Shortest path depiction : Diagonal

FOR FINAL RUN AND PLEASE HAVE A LOOK : http://roshanrai.me/hello-mars-/

The Aatmnirbhars!

R Sankalp Shukla

Roshan Rai

https://github.com/beardedghost4

https://github.com/kirito22037

Shikha Jha

https://github.com/shikha1810

Saurabh Rai

https://github.com/saurabhrai1998