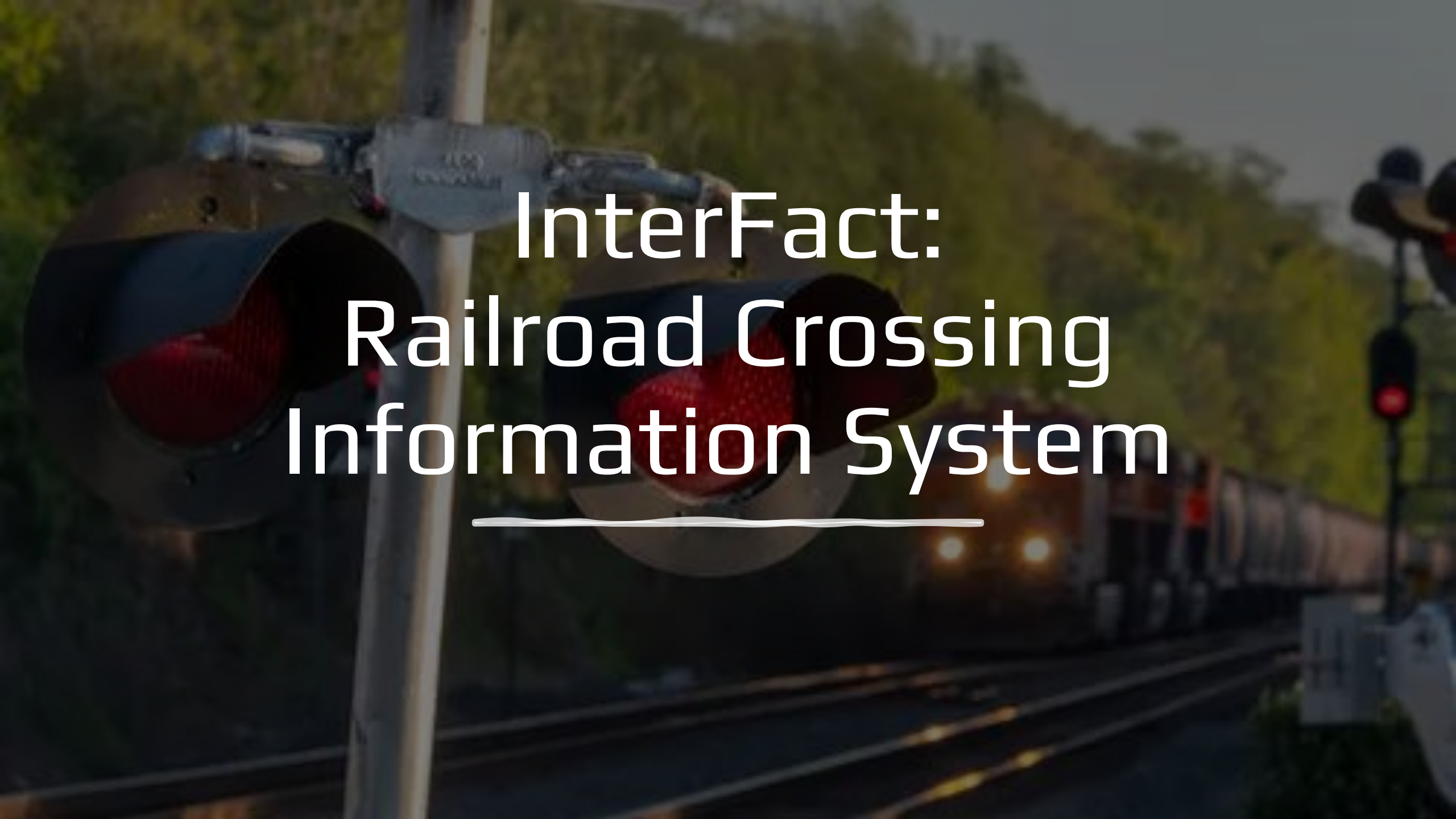


InterFact: Railroad Crossing Information System

InterFact Client & Information:

Huseyin Ergin

You know him

InterFact

Determines if a train is blocking an intersection in Muncie, using cameras and machine learning

An Admin Dashboard for reviewing data about the InterFact system

4th Iteration Features

- Interfact Admin Dashboard:
  - SQL Database Integration for User Reports
  - Intersection Prediction Data

Interfact Iteration 4 Demo!

MySQL User Reports Database

MySQL Database For User Requests!

```ts
import mysql from "mysql2/promise";

declare global {
  var dbMS: mysql.Pool | undefined;
}

const dbMS = mysql.createPool({
  host: process.env.DB_HOST as string,
  user: process.env.DB_USER as string,
  password: process.env.DB_PASS as string,
  database: process.env.DB_NAME as string,
  port: parseInt(process.env.DB_PORT || "3306"),
  waitForConnections: true,
  connectionLimit: 10,
  queueLimit: 0,
});

export default dbMS;
```

MySQL Config

Intersection Predictive Data!

- Supportive data for each intersection is now calculated:
  - Average amount of time an intersection is blocked
  - Total time an intersection is blocked (last 24 hours & week)
  - Normalized scoring system rating the intersection based on its blocked time

```
<div className="score-display">
    {hoveredHour !== null && hourlyScores[hoveredHour] !== undefined ? (
      <p>
        {hoveredHour < 12 ? times[hoveredHour] : times[hoveredHour - 12]}{" "}
        {hoveredHour < 12 ? "AM" : "PM"} has a blocked score of [ {hourlyScores[hoveredHour]} ]
      </p>
    ) : (
      <p> </p>
    )}
  </div>
```

```
useEffect(() => {
  if (logs.length > 0 && intersection) {
    const avgBlock = calculateAverageBlockageTime(logs, intersection.id);
    setAvgBlockTime(avgBlock)
  }
}, [logs, intersection]);
```

```
<div className='log-overall-time shadow'>
<div className='total-block-time'><h1>Average Time Blocked:</h1> <h2>{avgBlockTime} minutes</h2></div>
    <div className='total-block-time'><h1>Total Time Blocked (Last 24 hours):</h1> <h2>{blockedDayTime} minutes</h2></div>
    <div className='total-block-time'><h1>Total Time Blocked (Last Week):</h1> <h2>{blockedWeekTime} minutes</h2></div>
</div>
```

Ethan

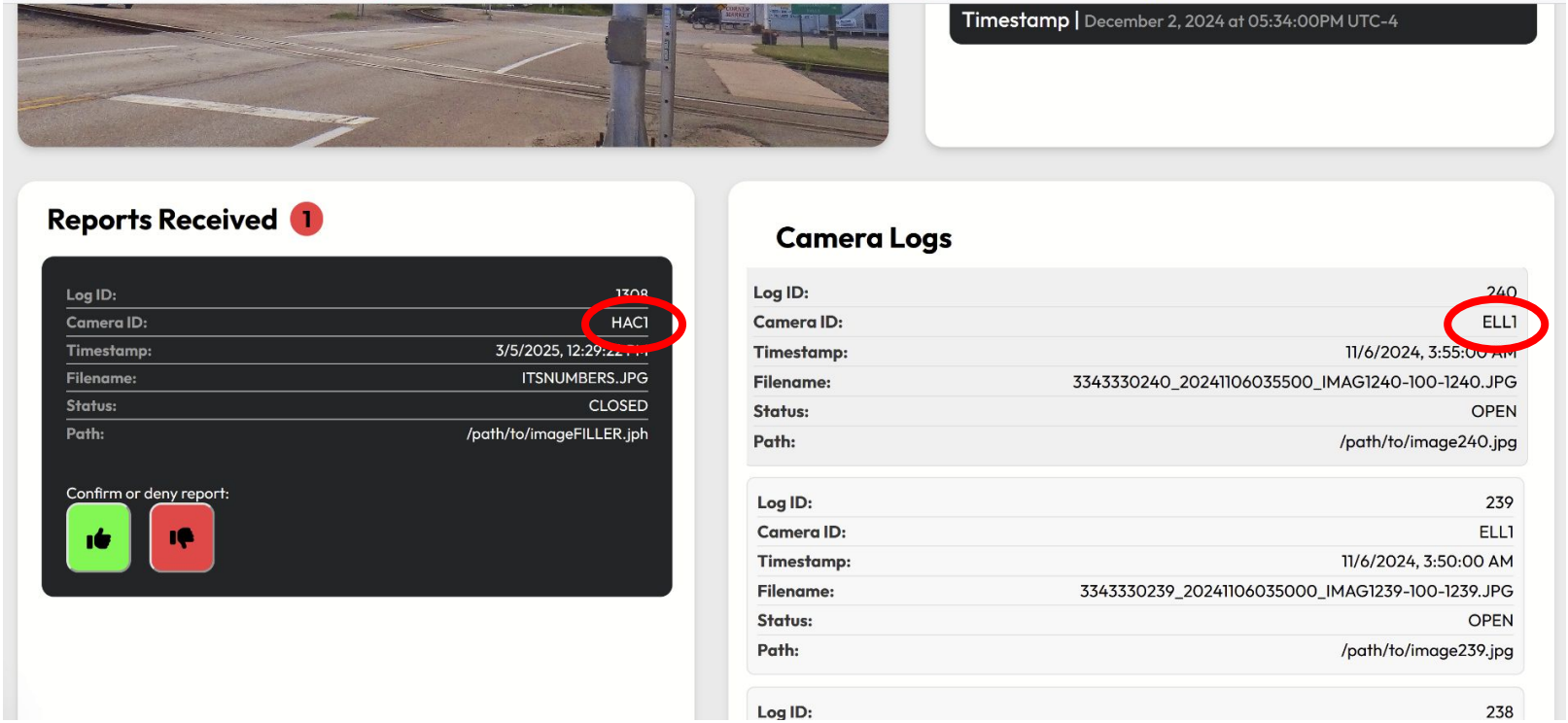
I worked on creating several data analysis widgets.

Brooke

File	% Stmts	% Branch	% Funcs	% Lines
All files	73	67.64	75	72.75

code coverage

I worked on testing the new features as well as fixing bugs the new features introduced.



bug showing reports for the wrong intersection

Tanner

I worked on making a transition when clicking on the links on top of the page to add more to the user experience.

```
<body ref={container}>
  <div className='transition'>
    <div className='transition-row row-1'>
      {[...Array(5)].map((_, i) => (
        <div key={i} className='block'></div>
      ))}
    </div>
    <div className='transition-row row-2'>
      {[...Array(5)].map((_, i) => (
        <div key={i} className='block'></div>
      ))}
    </div>
  </div>
```

```
function animateTransition() {
  return new Promise((resolve) => {
    gsap.set('.block', { visibility: 'visible', scaleY: 0 });
    gsap.to('.block', {
      scaleY: 1,
      duration: 1,
      stagger: {
        each: 0.1,
        from: 'start',
        grid: [2, 5],
        axis: 'x',
      },
      ease: ease,
      onComplete: resolve,
    });
  });
```

```
    document.querySelectorAll('a').forEach((link) => {
      link.addEventListener('click', (event) => {
        event.preventDefault();
        const href = link.getAttribute('href');

        if (href && !href.startsWith('#') && href !== pathname) {
          animateTransition().then(() => {
            router.push(href);
          });
        }
      });
    });

    revealTransition().then(() => {
      gsap.set('.block', { visibility: 'hidden' });
    });
  }, [isMounted, pathname]);

  if (isExcluded) {
    return (
      <html lang='en'>
        <body>{children}</body>
      </html>
    );
  }
```

Mason

This iteration, I worked on a few small usability features such as researching / using localStorage in Java to save preferences like filter presets. I also started working on implementing a dark theme and dark theme toggle using the same browser storage.

Additionally, I worked on creating the presentation slides and updating the documentation.

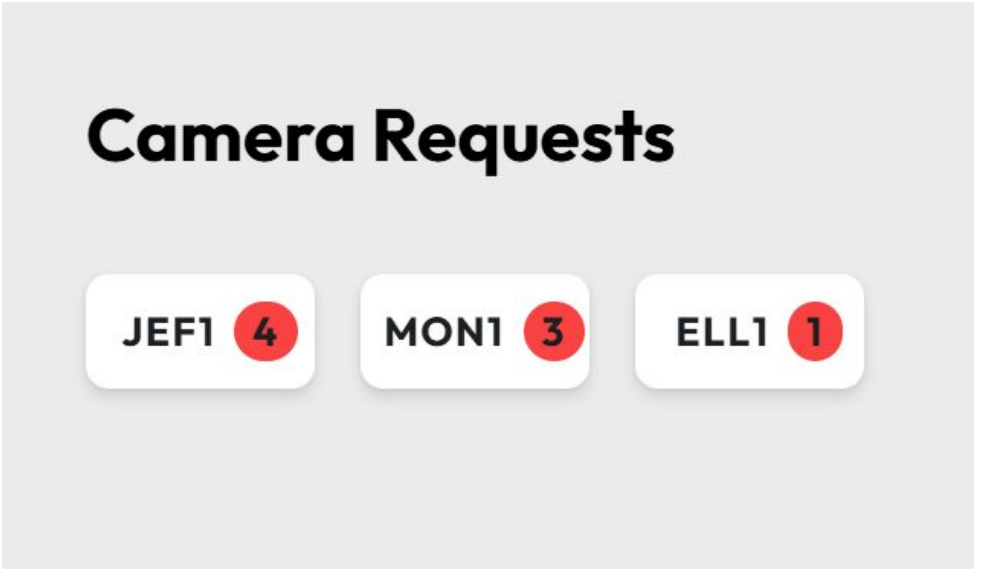
```
//------------------ Save Filter States to localStorage -----------------
// Will run whenever a filter is changed
useEffect(() => {
    // Store current filter states in localStorage whenever any of the filter state changes
    const filterStates = {
        isFiltering,
        isFilterOpen,
        isFilterBlocked,
        isFilterMaintenance,
        isFilterWorking,
        isFilterNotWorking
    };

    // Save the filter states in JSON string
    localStorage.setItem('filters', JSON.stringify(filterStates));
}, [isFiltering, isFilterOpen, isFilterBlocked, isFilterMaintenance, isFilterWorking, isFilterNotWorking]);
```

```
// ------------------------- Dark Theme -------------------------

// ---------------- Set Theme -------------
useEffect(() => {
    // Check localStorage if the theme was set previously
    const savedTheme = localStorage.getItem('theme');
    if (savedTheme === 'dark') {
        setIsDarkTheme(true);
        // Apply dark theme to body element
        // Might need to further specify additions if colors beyond background body are to be changed
        document.body.classList.add('dark-theme'); // Apply dark theme to body
    } else {
        setIsDarkTheme(false); // aka regular theme
        document.body.classList.remove('dark-theme'); // Remove dark theme from body element
    }
    // Empty array below makes sure this runs on load
}, []);
```

Bella



I worked on the requests page. Visually you see how a request is sorted by street id and how many user have requested that intersection.

```
6
7   export default function requests() {
8
9       const userFeedback = useUserFeedback();
10      const params = useParams();
11      const [requests, setRequests] = useState<string[] | null>([]);
12      const id = Array.isArray(params.id) ? params.id[0] : params.id;
13
14      const getRequests = (id : string): string[] => {
15          return userFeedback.flatMap(user => {
16              if (Array.isArray(user.requests)) {
17                  return user.requests.filter((request : string) => {
18                      return request === id
19              });
20              }
21              return [];
22          });
23        };
24
25      useEffect(() => {
26      if (userFeedback.length > 0 && id) {
27          const fetchedRequests = getRequests(id);
28          setRequests(fetchedRequests);
29      }
30      }, [userFeedback, id]);
```

Mentor Feedback

- Feedback :
  Bradley told us the new features look good functionally but the ui design could use a little updating to something more like a bar chart.

  He also told us it would be ideal to only render a few logs at a time and have a page system to avoid lagging due to the amount of logs the program was trying to load at one time.

Client Feedback

- Feedback:
  - Would like to transition from normalized score to percentage. i.e. How likely is the intersection going to be blocked at a given time of day?
  - New features are usable and show potential.
- Changes Made:
  - Added notes & changes to Iteration 5 goals.

Planned Iteration 5 Features:

- More detailed prediction display data
  - Transition normalized score to blocked percentage
  - Break up single 24 hour visual into visuals for each day of the week

- Implement catch for average calculation for missing log reports.