Deliverable 4 - Dev					
Risk	**Score**	**Severity (1/5)**	**Likelihood (1/5)**	**Mitigate**	**Contingency**
1. Dropping class	5	5	1	- be respectful to group members - encourage group members	- have someone shadow high priority roles
2. Changing requirements	6	3	2	- meet with stakeholder frequently	- chosen hosting platform: Firebase allows easy database integrations if requirements change
3. Lag with datasets	8	4	2	- use small datasets - test early with dataset to observe need of optimizations	- create a server using node.js or other server language to offload
4. Inadequate documentation	6	3	2	- review code (methods and classes) before pushing to codebase - follow strict documentation rules	- refactor for self explanatory code
5. Poor communication	3	3	1	- discord always available - discussions frequently	- use discord for emergency meeting
6. Learning new tech stack (gherkin, webxr, etc.) takes up time, higher chance of mistakes	6	3	2	- open communication between dev and test team to share what they know and help each other out - share where to learn this new tech stack	- prepared to help each other - walk group member through their problem - take on another task
7. Scope creep	3	3	1	- become familiar with requirements - regular meetings with stakeholder	- communicate if you notice some scope creep and plan for redirection
ID2 - UPDATED SCORE					
8. Midterm exams	0	0	0	- know everyones schedule - plan around midterms	- have someone ready to take over or help complete task
9. CSPIP interviews (time conflicts amongst team)	0	0	0	- know everyones schedule - plan around midterms	- have someone ready to take over or help complete task
10. Possible incompatability with modules/libraries (ex. Jest + Drei Text component)	12	4	3	- research how to set up and use libraries/modules - test that libraries/modules work together	- have someone who set up libraries/modules available to help or fix problems
11. indexedDB may overflow browser memory	4	4	1	- research and test indexedDB if it can be used	- pivot to a server - drop columns from data point instantiation before quering indexedDB
12. Managing security/permissions for pipelines to avoid downtime	8	4	2	- staggered pipeline deployment - regularly update and maintain pipeline - multiple reviews before Git commit to branch	- update security/permissions as needed during downtime
13. Limited early testing/debugging capabilities	16	4	4	- test as best as you can with unit tests - robust logging to find errors - use assertions	- prioritize debugging - allow for roll backs
ID3 - NEW RISKS					
14. ESLint not working correctly	3	3	1	- run npm run lint --fix command to fix any style issues before pushing any commits	- refer to the Airbnb + React dcoumentation to improve ESLint
ID4 - NEW RISKS					
15. IndexedDB data getting jumbled	10	5	2	- be cautious that data in a row isn't mismatched across columns, so double check the rows in the database are correct	- do some refactoring to adjust mismatching
16. Delayed implmentaion of DAL, so issue dependencies are also delayed	16	4	4	- prioritze the DAL	- prioritize implementing the DAL

Deliverable 4 - Test					
Risk	**Score**	**Severity (1/5)**	**Likelihood (1/5)**	**Mitigate**	**Contingency**
1. Dropping class	5	5	1	- be respectful to group members - encourage group members	- have someone shadow high priority roles
2. Code changes close to deadline	8	4	2	- put in strict deadlines	- shadow dev team to prepare for what tests may be needed
3. Unoptimized code	6	3	2	- testing to find the unoptimized code	- prioritize refactoring code
4. Inadequate documentation	6	3	2	- review code (methods and classes) before pushing to codebase - follow strict documentation rules	- refactor for self explanatory code
5. Poor communication	3	3	1	- discord always available - discussions frequently	- use discord for emergency meeting
6. Learning new tech stack (gherkin, webxr, etc.) takes up time, higher chance of mistakes	6	3	2	- open communication between dev and test team to share what they know and help each other out - share where to learn this new tech stack	- prepared to help each other - walk group member through their problem - take on another task
7. Manual integration test plan	12	4	3	- research automatic testing - test different automatic testing tools	- well document the manual testing - clean test code - create proper test cases with manual testing
ID2 - UPDATED SCORE					
8. Midterm exams	0	0	0	- know everyones schedule - plan around midterms	- have someone ready to take over or help complete task
9. CSPIP interviews (time conflicts amongst team)	0	0	0	- know everyones schedule - plan around midterms	- have someone ready to take over or help complete task
10. Jest not performing as expected for unit tests (ex. Jest + Drei Text component)	12	4	3	- research how to set up and use Jest	- have someone who set up Jest available to help or fix problems
11. indexedDB may overflow browser memory	4	4	1	- research and test indexedDB if it can be used	- pivot to a server - drop columns from data point instantiation before quering indexedDB
12. Managing security/permissions for pipelines to avoid downtime	8	4	2	- staggered pipeline deployment - regularly update and maintain pipeline - multiple reviews before Git commit to branch	- update security/permissions as needed during downtime
13. Limited early testing/debugging capabilities	16	4	4	- test as best as you can with unit tests - robust logging to find errors - use assertions	- prioritize debugging - allow for roll backs
14. Incosistency with style for gherkins because ESLint does not check gherkins	6	2	3	- follow a style guide for creating gherkins in wiki	- refactor gherkin files
15. Secuity risk if logger leaks sensitive information onto another server out of our control	10	5	2	- ensure restrictions are put in place, only send to "this spot" - test with non important info first to see if logging info is lost	- send logger info to a back up location