<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Username</th>
</tr>
</thead>
<tbody>
<tr>
<td>Katie Pfleger</td>
<td>Co-Manager</td>
<td>kjp2157</td>
</tr>
<tr>
<td>Julia Sheth</td>
<td>Co-Manager</td>
<td>jns2157</td>
</tr>
<tr>
<td>Alana Anderson</td>
<td>Language Guru</td>
<td>afa2132</td>
</tr>
<tr>
<td>Nicholas Sparks</td>
<td>Tester</td>
<td>ns3284</td>
</tr>
<tr>
<td>Pearce Kiesser</td>
<td>System Architect</td>
<td>pck2119</td>
</tr>
</tbody>
</table>
Machine learning is now more prevalent than ever, with 51% of enterprises deploying ML.

Matrix manipulations are often essential to machine learning algorithms.
PROJECT WORKFLOW: TOOLS

- docker
- OCaml
- THE C
- PROGRAMMING LANGUAGE
PROJECT WORKFLOW: TIMELINE

Sep 9, 2018 – Dec 15, 2018

- **sept. 19th**: project proposal
- **oct. 15th**: LRM, scanner, parser
- **nov. 14th**: hello world
- **nov. 17th**: C library
- **nov. 27th**: adding matrix to codegen
- **dec. 15th**: matrix operations working
- **dec. 17th**: present!
- **dec. 14th**: print matrix
LANGUAGE OVERVIEW:

operators
+, -, *, /, %, >, <,>=, <=, ==

control flow
for (i = 0; i < 5; i = i + 1) {}
while (i > 5) {}
if(i == true) {} else {}

comments
/* this is a comment in MATRX */

non-matrix types
int, bool, void, float, string, char

operators
+, -, *, /, %, >, <,>=, <=, ==

function declarations:
int main() { return 0; }
matrix foo(matrix m) { return m; }
matrix declaration:
matrix m; /* declares a matrix m */

matrix initialization:
matrix m = [[[1,2][3,4]]]; /* declares and initializes m */
matrix n; /* declares a matrix n */
n = [[[5,6][7,8]]]; /* initializes n */
matrix arithmetic operations:

matrix m = [[1,2][3,4]];
matrix n = [[5,6][7,8]];
printm(transpose(m)); /* prints the matrix m transposed */
printm(matmult(m,n)); /* multiplies matrix m by matrix n */
printm(matadd(m,n)); /* adds matrices m and n */
printm(dot(m,n)); /* takes the dot product of m and n */
printm(det(m,2)); /* takes the determinant of m, which has dimensions 2 x 2 */
We used a C library to implement matrix functions and then linked the library through codegen.

```c
struct matrix {
    int num_rows;
    int num_cols;
    int** matrixAddr;
    int buildPosition;
};
```
TESTING:

- At any given point, each new feature in codegen is semantically checked.
- Used regression test suite with target pass/fail test cases, ensure that other features still worked.
- If necessary, perform manual checks.
DEMO
THANK YOU!

SPECIAL THANKS TO OUR TA DEAN DENG!