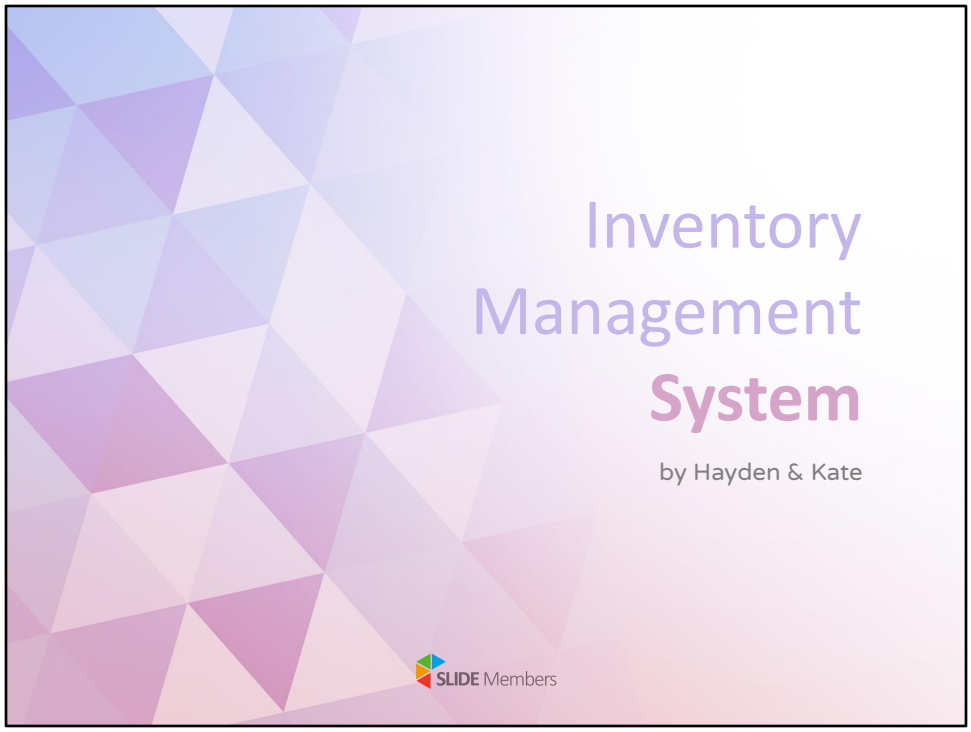
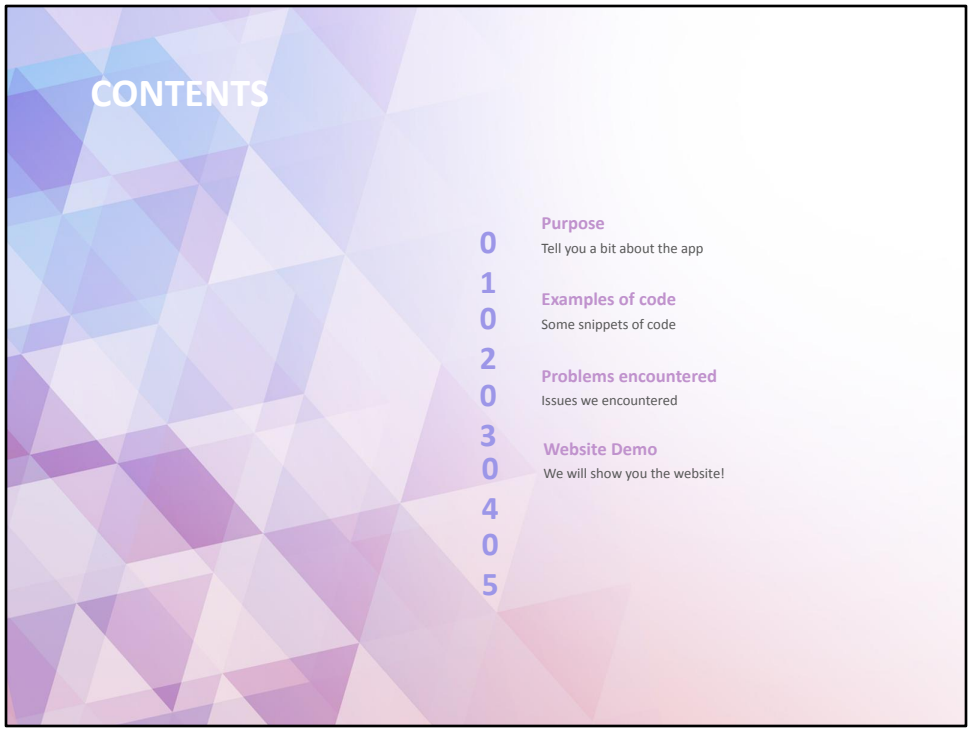


Welcome to our presentation,
so Hayden & I built an app called Inventory Management System.



In this presentation, We will tell you all about our app, share some code snippets, some problems we encountered and finally we will do a demonstration of our website.

Inventory Management System

What it does:

This app allows either a manager or employee of a store to sign up, log in and view an inventory of their products.

Functionality:

- *Sign up; as a manager or employee*
- *Log in; as a manager or employee*
- *Create, update or delete products*
- *Edit; products name, price, quantity and category*
- *Filter; products by name, price, quantity, category or id*
- *Backend: Managers can delete users but employees can't*
    - *this hasn't been implemented into the front end*

Example of a function:

src > functions > JS jwtFunctions.js > validateUserAuth

```javascript
28
29  async function validateUserAuth(request, response, next){
30      console.log(request.headers);
31      const { authorization, jwt } = request.headers;
32
33      console.log(authorization, jwt)
34
35      if (!authorization && !jwt){
36          return response.status(403).json({
37              message:"No authorization token provided."
38          });
39      }
40
41      let providedToken;
42
43      if (authorization) {
44          const [,token] = authorization.split(' ');
45
46          providedToken = token;
47      } else if (jwt) {
48          providedToken = jwt;
49      }
50
51      console.log(providedToken);
52
53      let decodedData = decodeJWT(providedToken);
54      console.log(decodedData);
55      if (decodedData.userId){
56          request.authUserData = decodedData;
57          next();
58      } else {
59          return response.status(403).json({
60              message:"No authorization token provided."
61          });
62      }
```

validateUserAuth:

Checks if the user is logged in by

getting the token out of the jwt or authorization header, (t splits the authorization header and takes just the token part)

then it decodes the jwt to check the identity

If not logged in,

return message “No authorization token provided”

KATE: Here I have a snippet of code from the backend, from the jwtFunctions file specifically, where I have a function that checks if the user is logged in with an error message being returned if they are not (logged in)

we were having issues with the jwt header so we made it so it accepts either authorization bearer or jwt token

Example of conditional statement

src > controllers > JS ProductController.js > router.patch("/:id") callback

```
// Update Product by ID
router.patch("/:id", validateUserAuth, async (req, res) => {
    // Expects updateData in the request body
    const updateData  = req.body;

    try {
        let product;
        const id = req.params.id;
        // Find product by either ID or item
        if (id) {
            product = await updateOneProduct(id, updateData);
        }
        // if product is not found message
        if (!product) {
            return res.status(404).json({
                success: false,
                message: "Product not found to update",
            });
        }
        // if product found and updated message
        res.json({
            success: true,
            message: "Product updated successfully",
            data: product,
        });
    // Error message
    } catch (error) {
        console.error(error);
        res.status(500).json({
            success: false,
            message: "Error updating product",
            error: error.message,
        });
    }
});
```

Peep the previous function mentioned being called here

If product is not found,
Return error message of
“Product not found to update”

KATE: Here is an example of a conditional statement, from the ProductController. In this section of code, it is getting the ID of the product and then allowing us to update the file, with conditional statements saying if product is not found, return a error message.
If product IS found and updated, it also returns a message.
You might also notice that the vaidateUserAuth function, which i mentioned in the previous slide, is being called at the top here also, to ensure that the user is logged in before updating the product.

Example of database operations:

Create Product function in ProductCrud.js

src > utils > crud > JS ProductCrud.js > ...

```javascript
// Provide CRUD functions for the ProductModel
const { query } = require("express");
const { ProductModel } = require("../../models/ProductModel");

// Create new product
async function createProduct (product) {
    //name, price, quantity, category, description = null) {

    let result = await ProductModel.create(product);
        // name: name,
        // price: price,
        // quantity: quantity,
        // category: category,
        // description: description
    return result;
}
```

KATE: Database operation that create products
I created product object with the following things

More examples of database operations:

Update product

```
// Update one product
async function updateOneProduct(id, updateData) {
    return result = await ProductModel.updateOne({_id: id}, updateData);
}
```

Delete product

```
// Delete one product
async function deleteOneProduct (id) {
    return result = await ProductModel.findByIdAndDelete(id);
}
```

KATE: Some more examples we have a function that updates the product and one that deletes the product.

Example of network request:

```
const handleSubmit = async (e) => {
    e.preventDefault();
    try {
        const response = await axios.post(
            "https://ims-backend-2qfp.onrender.com/products/create",
            formData
        );

        if (response.status === 201 || response.status === 200) {
            togglePopup();
            setFormData({ name: "", price: "", quantity: "", category: "" });
            fetchItems();
        }
    } catch (error) {
        console.error("Error adding item:", error);
    }
};
```

Creating Products on the front-end.

HAYDEN:

Problems encountered:

- *Price validator*
  - *tried to format the Price number into a monetary value*
  - *the validator I found online didn't work*
  - *decided to render in the front-end instead.*
- *Git merges*
  - *had conflicts merging,*
  - *put restrictions in place that meant a pull-request had to be required before merging to main*
- *the dreaded misplaced comma*
  - *forgot a comma in one of the database operations*
  - *code wouldn't work until rectified*
- *JWT/Auth issues*
  - *edit & delete wouldn't work*
  - *so made it so it accepted both JWT OR Authorization Bearer*

KATE:

Time to check out the website!

https://ims-hk.netlify.app/

KATE: I'll turn it over to Hayden now and he can demo the website!
HAYDEN:

THANK
YOU