UTS: ENGINEERING & INFORMATION TECHNOLOGY

SUBJECT NUMBER & NAME	**NAME OF STUDENT(s)** (PRINT CLEARLY)	**STUDENT ID(s)**
41025: Introduction to Software Development	Alice Nguyen	12625582
	Marina Santanelli	13558933
	James Smith	13254091
	Kayla Gelman	13555115

STUDENT EMAIL	**STUDENT CONTACT NUMBER**
12625582@student.uts.edu.au	0432159921
13558933@student.uts.edu.au	0452279151
13254091@student.uts.edu	0432222586
13555115@student.uts.edu.a	

NAME OF TUTOR	**TUTORIAL GROUP**	**DUE DATE**
Muhammad Atif Qureshi	Tutorial 3, Tuesday 6PM, Group 14	17/05/2021
Dr Atif Qureshi		

ASSESSMENT ITEM NUMBER & TITLE

Assessment 2 – Implementation & Testing

☒ I confirm that I have read, understood and followed the guidelines for assignment submission and presentation on page 2 of this cover sheet.

☒ I confirm that I have read, understood and followed the advice in the Subject Outline about assessment requirements.

☒ I understand that if this assignment is submitted after the due date, it may incur a penalty for lateness unless I have previously had an extension of time approved and have attached the written confirmation of this extension.

Declaration of originality: The work contained in this assignment, other than that specifically attributed to another source, is that of the author(s) and has not been previously submitted for assessment. I understand that, should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with University policy and rules. In the statement below, I have indicated the extent to which I have collaborated with others, whom I have named.

Statement of collaboration:

Signature of student(s)

[illegible] [illegible] [illegible] [illegible]

Table of Contents

1. Individual Student Contribution

ID	Feature	Create	Read	Update	Delete	Responsible
01	Online User Access Management [MVC] This applies to both customer and staff user.	A user (e.g. customer, staff) can sign up online (full name, email [as a username], password, phone) A registered user access logs (user id, login date/time, logout date/time) are stored in the database.	A registered user can view their registration detail. A registered user can login and logout from the system. A registered user can list (view) their access logs and search the log records based on the date.	A registered user can update their registration details. A registered user cannot update their user access logs.	A registered user can cancel their registration. A registered user cannot delete their user access logs.	James
02	IoT Device Catalogue (Collection) Management [MVC]	Only staff user can create the IoT device (product) details in the database. i.e. device name, type, unit price, quantity (stock).	Customer and staff user can list the IoT device records. Customer and staff users can search and list the devices based on their name and type.	Only staff user can update the saved IoT device records.	Only staff user can delete the saved IoT device records.	Kayla
03	Order Management [MVC]	A customer user (registered or anonymous) can create (save, submit) an order for IoT devices.	A customer user can view their saved order details, order history list, and search the orders based on the order number and date.	A customer user can update their saved order before final submission.	A customer user can cancel their saved order before final submission.	Alice
04	Payment Management [MVC]	A customer user can create payment details (payment method, credit card details, amount, date) for their order (payment id linked to order id).	A customer user can view their saved order payment details, payment history list and search the specific payment records based on the payment id and date.	A customer user can update their saved payment details record before finalising the payment.	A customer user can delete their saved payment details record before payment submission	Marina

2. Working Software Code (MVC)

GitHub link: https://github.com/wowaleece/IoTBay

3. Feature Documentation

3.1. Online User Access Management [MVC] – 01

The online user access management system executes the operations that allow users to create an account, login, logout, view logs and everything related to user management. As stated in section 1 – the individual statement contribution outline, the purpose of the feature is to create, read, update and delete users from the system. However, all delete and some update commands must be Admin managed.

3.1.1. Functional Requirements

The following table outlines how user stories from assessment 1 are mapped against the CRUD operations implemented in our final IoTBay system.

USR ID	Description	Create	Read	Update	Delete
USR102	As an admin, I want to be able to login to the online application using my registered User ID	N/A	The database uses an authentication function, allowing an admin to log in.	N/A	N/A
USR104	As a customer, I want to be able to register into the online application using my User ID (email) and password	Upon registration, a form is created and submitted that validates the users registration.	N/A	N/A	N/A
USR105	As a customer, I want to be able to login to the online application using my registered UserID.	N/A	A customers' entered email and password is validated against the current database.		N/A
USR107	As a user (customer and admin), I want my account to be restricted by a password.	When creating an account, password is a protected field.	N/A	N/A	N/A

USR ID	Description	Create	Read	Update	Delete
USR110	As an admin, I want to create users.	When registered as an admin, multiple users can be manually created.	N/A	N/A	N/A
USR111	As an admin, I want to Read users.	N/A	All registered users are read from the USERS database and displayed to the admin.	N/A	N/A
USR112	As an admin, I want to update user information	N/A	N/A	An admin has the option to edit user information.	N/A
USR113	As an admin, I want to delete users	N/A	N/A	N/A	An admin can delete users from the database.
USR114	As an admin, I want to View application access logs	Time stamp logs are created when a user logs into or out of the system.	Application logs are displayed in admin view.	N/A	N/A
USR202	As customer, I want to change login password	N/A	N/A	A user is able to login and edit their profile	N/A

3.1.1.1. Changes

USRID	Description	Change	Reason
USR101	As an admin, I want to be able to register into the online application using my User ID (email) and password	Not implemented in R1 release.	This user story is now out of scope as a super admin should be able to add other admins rather than allowing a user to apply to be an admin.
USR108	As a user (customer and admin), I want my password to be hashed and salted	Postponed to R2 Release.	This user story is now out of scope due to time constraints.

3.1.2. Non-Functional Requirements

The non-functional requirements specific to online user management include the performance of the system, Security, usability and data quality.

The primary non-functional requirement was that only hashed passwords would be stored securely and cut off from the world. We have a technical knowledge gap to bridge for implementing this, therefore it has been pushed back to a later release, and instead passwords were simply stored with the user. The current spike implemented to support a move to hashed passwords later.

3.1.3. Acceptance Testing

Test ID	USR ID	Test Criteria	Pass/Fail	Comment
TID001	USR102	Admin is directed to a unique admin view	Pass	
TID002	USR104	User is able to successfully register an account which updates the database	Pass	
TID003	USR105	Customer is able to login using registered userID successfully.	Pass	
TID004	USR106	Customer is able to logout within 2 clicks	Failed	Errornous & currently unreliable
TID004	USR107	Password is hashed and salted	Failed	Hashing algorithm is implemented when processing the password
TID005	USR110	Admin is able to create a new user	Failed	Not in scope for minimum viable product
TID006	USR111	Admin is able to view all registered users	Failed	Not in scope for minimum viable product
TID007	USR112	Admin is able to update user information	Failed	Not in scope for minimum viable product
TID008	USR113	Admin is able to delete users and query is reflected in the database	Failed	Not in scope for minimum viable product

TID009	USR114	Customer is able to view application logs	Pass	
TID010	USR202	Customers update details is reflected in the database	Pass	

3.1.4. Defect Log

DefectID	Description	Date	Test Case ID	Tester	Responsible	Status	Comments
DID001	Access logs don't record when someone clicks off a tab.	16/05/2021	USR114	James	James	In progress	Need learn about onBeforeUnload() or similar function
DID002	Logout servlet error	16/05/2021	USR106	James	James	In progress	Needs better flow control.
DID003	Edit customer does not update bean, although does update the database.	16/05/2021	USR202	James	James	In Progress	Retrival of Customer's full address requires better input validation.
DID004	Logs not appearing in included statement	16/05/2021	USR114	James	James	resolved	Resolved flow control issues
DID005	Admin filter not applying	16/05/2021	USR102	James	James	resolved	Resolved WEB.XML config
DID006	Register user login but not customer details	16/05/2021	USR202	James	James	resolved	Refactored code to delete users if customer could not be added, will need to refactor again with SQL "Transactions"
DID007	User Login not getting Customer details	16/05/2021	USR105	James	James	resolved	

3.2. IoT Device Catalogue (Collection) Management [MVC]

The purpose of the IoTDevice Catalogue (collection) management system is to create, update, read, delete and show all products in the IoTBay inventory.

3.2.1. Functional Requirements

The following table documents the CRUD mapping to the original user stories. The original table with all user stories can be found in the appendix.

Provide a brief functional description of the assigned software application feature or module and its mapping to user stories (CRUD). Indicate any changes in the user stores/design etc. since assignment.

USR ID	Description	Create	Read	Update	Delete
USR302	As a customer, I want to be able to view IoTBay products online at any time	N/A	All products are displayed when a user chooses to browse the inventory.	N/A	N/A
USR303	As a customer, I want to browse IoTBay products	N/A	All products are displayed when a user chooses to browse the inventory.	N/A	N/A
USR304	As a user, I want to be able to type in a search bar.	N/A	A query is registered when a user types in the search bar to read the product (if existing) in the database.	N/A	N/A
USR702	As an IoT staff member, I want to track inventory levels	N/A	When registered as an admin, an IoT staff member is able to view the stock level of each product.	N/A	N/A
USR703	As an IoT Staff member, I want to restock Inventory / update	An IoT Staff member is able to create new product to be added	All inventory items are read so that an IoT staff member can choose	All inventory products are able to be updated.	All inventory products are able to be deleted with Admin permissions.

	inventory quantities.	to the inventory.	which item to update.		

3.2.2. Non-Functional Requirements

The key non-functional requirements discussed in Assessment 1 include performance and security of the application, these requirements can be found in the appendix. Performance user stories #USR306 and #USR901 were both achieved as a both a customer and IoTBay staff member can receive a research result within a minute and have a page (in this case the display of products) returned within a minute of making a request.

3.2.3. Acceptance Testing

Test ID	USR ID	Test Criteria	Pass/Fail	Comment
TID001	USR302	IoTBay can be accessed anytime and by anyone	Fail	IoTBay is still In Beta testing and therefore, can only be accessed through the netbeans server or stored on your local device.
TID002	USR303	Customers can only view products in the database and cannot add/edit information.	Pass	
TID003	USR304	A user can easily search for inventory in the system.	Partial	A user is able to type in a search bar but it fails to process the request to the servlet.
TID004	USR702	An admin has their own inventory view that shows the total quantity of products remaining.	Pass	
TID005	USR703	Only an admin is able to Update product details or add new ones.	Pass	

3.2.4. Defect Log

DefectID	Description	Date	Test Case ID	Tester	Responsible	Status	Comments
DID001	CRUDS commands are not executing.	17/05/2021	TID002 TID004 TID005	Kayla	Kayla	Resolved	Build failed for running TestProductsDB
DID002	Unable to connect ViewProducts.jsp to servlet.	07/05/2021	TID002	Kayla	Kayla	Resolved	Webxml connection must be set up and database CRUD commands must be correct in order for the file to run.
DID003	Cannot create product in TestProductsDB file	04/05/2021	TID003	Kayla	Kayla	Resolved	Error was caused due to incompatible variables passed into DBProducts.
DID004	Unable to view data from servlet – 500 Internal Error	07/05/2021	TID002 TID005	Kayla	Kayla	Resolved	After 2 weeks, this issue was solved. The issue due to the structure of the servlets missing the web.xml extension.
DID005	Failed to run program.	14/05/2021	TID002 TID005	Kayla	Kayla	Resolved	After adding a servlet to view all products, the system completely broke and would not let me test any files. The solution was to repull most up to date branch from git.
DID006	Search results are not displayed from servlet	15/05/2021	TID003	Kayla	Kayla	In Progress	Model is set up, however, the servlet won’t connect to the button in order to execute the command.
DID007	Delete button is not connecting to servlet – 500 Internal error	15/05/2021	TID003	Kayla	Kayla	In Progress	CRUD test file works when deleting the product
DID008	Search results are only specific for Producs	15/05/2021	TID003	Kayla	Kayla	In Progress	Need to set up a function to display the product based on category.
DID009	Any uses (customers or staff) are able to view the admin view.	15/05/2021	USR703	Kayla	Kayla	Resolved	Implemented a command to limit non-admins visibility.

3.3. Order Management [MVC]

3.3.1. Functional Requirements

The aim of this feature is to manage orders of multiple users of the system. This allows users to view their order, track their order status, cancel or update their order if they change their mind and save their orders to process for next time. Admins are also able to make and process changes to order details, including creating, reading, updating and deleting orders and order line items.

USR ID	Description	Create	Read	Update	Delete
USR305	As a customer I want to be able to add products to my shopping cart	Orders are created upon adding items to shopping cart.	N/A	N/A	N/A
USR307	As a customer I want to be notified of a successful order after it being placed	N/A	N/A	Once orders are updated and confirmed, user is redirected to a page with an updated status message.	N/A
USR501	As a customer I want to receive an email notification once I have placed an order	N/A	N/A	Once orders are updated and confirmed, user receives an email notification of confirmation.	N/A
USR502	As a customer I want to receive an order number	Order numbers are created upon adding items to shopping cart and are given upon saving of order.	N/A	N/A	N/A
USR503	As a customer I want to track my order using my order number	N/A	Customers are able to search via order number to view their order details.	N/A	N/A
USR504	As a customer I want to view purchase history	N/A	Registered customers are able view their entire order history.	N/A	N/A

3.3.2. Non-Functional Requirements

3.3.2.1 Changes

USR ID	Description	Create	Read	Update	Delete
USR507	As a customer I want to view my order within 10 seconds.	N/A	System should be efficient in processing operations to filter through orders numbers.	N/A	N/A
USR508	As a customer I want to receive an email notification within 30 minutes of a successful order.			System must be efficient in processing email requests.	

3.3.3. Order Management User Stories

3.3.3.1. Acceptance Testing

Test ID	User Story #	Test Criteria	Pass/Fail	Comments
T301	USR305	Customers can view and add selected products to shopping cart view.	Fail	View of cart currently does not connect to session data from products view and due to time constraints have not been implemented at the end of this iteration.
T302	USR307	Customers will be redirected to a successful order view page upon submitted their order.	Fail	As this feature is dependent on USR305 which also failed, this has not been implemented this iteration.
T303	USR501	Customers receive an email after successful order has been created.	Fail	As this feature is dependent on USR307 this feature has not been implemented.
T304	USR502	Customer has an order number generated upon saving or completing their order.	Pass	
T305	USR503	Customer is able to filter through and find their order via order number.	Partial	Have not completed view of this feature. Model and Controller complete.
T306	USR504	Customer can view their entire purchase history when logged in.	Partial	User can view entire purchase history however, is not

				limited to specific customers.

3.3.3.2. Defect Logs

DefectID	Description	Date	Test Case ID	Tester	Responsible	Status	Comments
DID301	CRUDS commands are not executing. Error: Build Failed	04/05/2021	T304 T305 T306	Alice	Alice	Resolved	Build failed for TestOrdersDB.java. Resolved after fixing notation error.
DID302	Conversion error of datatypes when running SQL queries. E.g. Error: incompatible types: java.lang.String cannot be converted to java.sql.datetime	04/05/2021	T304 T305 T306	Alice	Alice	Resolved	Build failed, execution error due to incorrect datatype parsing.
DID303	Controller testing failed to read next line from terminal. No error message displayed.	04/05/2021	T304 T305 T306	Alice	Alice	Resolved	Terminal would blink without calling next line in function.
DID304	Unable to connect to Servlets from JSP pages.	16/05/2021	T301 T302 T303	Alice	Alice	Resolved	URL mapping missing from web.xml
DID305	Order Views not displaying.	16/05/2021	T301 T302 T303	Alice	Alice	In Progress	Connection issue when calling Servlets from respective JSP pages.
DID306	Servlet not found – 500 Internal Error.	16/05/2021	T301 T302 T303	Alice	Alice	Resolved	Null Pointer Exception resolved

3.4. Payment Management [MVC]

3.4.1 Functional Requirements

The payment management feature of the web application is aimed at capturing user's payment details so that they can generate a transaction and purchase the items in their shopping cart successfully. Within this feature, a user should be able to create payment methods, or delete and edit them as they please.

USR ID	Description	Create	Read	Update	Delete
USR203	As a customer I want to change my payment details so that I can make sure my bank account details are up to date.	N/A	N/A	Payment details can be updated during payment process.	N/A
USR401	As a customer I want to enter my payment details so that I can place an order.	Payment details are captured by the payment process, at the start of the transaction.	N/A	N/A	N/A
USR402	As a customer I want to save my payment details so that I can use them for future transactions.	N/A	User's payment details are logged into database once entered into the payment details form.	N/A	N/A
USR403	As a customer I want to update my personal (payment) details so that future transactions use the correct details.	N/A	N/A	Payment details can be updated during payment process.	N/A
USR404	As a customer I want to have saved payment details entered into new orders so that I can save time placing subsequent orders.	N/A	User's payment details are logged into database once entered into the payment details form, and retrieved accordingly when a user logs in and starts a new transaction.	N/A	N/A
USR405	As a customer I want to receive an invoice so that as a guest, I am able to track my purchases.	Email notification is sent to designated user email.	N/A	N/A	N/A

USR502	As a customer I want to receive an order number so that I can track my order.	N/A	Order number appears during order confirmation page (and within invoice in email).	N/A	N/A
USR506	As a customer I want to view my purchase history so that I can track my order(s).	N/A	In later iterations, customers would be able to view their purchase history in their account logs.	N/A	N/A

Changes (since assignment 1):

USRID	Description	Change	Reason
USR505	As a customer I want to save shipment details so that I can track my order/delivery.	Postponed to R2 Release.	This user story is now out of scope due to time constraints.
(NEW)	As a customer I want to be able to delete my saved payment details before payment submission.	New user story added into feature.	This user story was vital to fulfil CRUD operations and allows users to cancel their payment

3.4.2 Non-Functional Requirements

Existing non-functional requirements relating to the payment management feature involves:

Performance:

- *USR901* – As a customer I want to have a page returned within 1 minute of my request, so that I am not frustrated while using the application.

Changes to non-functional requirements (since assignment 1):

Non-functional requirement	Description	Reason
	ADDED	
Validity	As a user I want to receive validation errors that provide me with a notification/banner, so that it is clear when there is an error and I need to change something in the form to fit the data standards for the web application.	Ensures valid data is entered into the system; enriching data integrity.
Performance	As a user I want my transaction to process within 10 seconds, so that I can make purchases quickly on the web application.	Improves user productivity and invites user to browse the site for longer.
Accessibility	As a user I want to view a consistent styling scheme so that I am not visually overwhelmed and won't experience eye strain.	Generates a better experience for users while using the web application.

	As a user I want features and processes to be simple and clear to use so that it is easy to navigate through the web application.	Generates a better experience for users while using the web application.
	As a user I want to view easily identifiable buttons and interactive items so that I know what to click on in order to navigate easily through the web application.	Generates a better experience for users while using the web application.
REMOVED		
Security – USR405	As a customer I want to use HTTPS when making a payment so that my payment details are encrypted in case of security attacks.	This non-functional requirement was not added due to it being out of scope.

3.4.3 Payment Management User Stories

Acceptance Tests:

Test ID	User Story #	Test Criteria	Result	Comments
T111	USR203	Customers should be able to change payment details so that they can make sure their bank account details are up to date.	Passed	Customers can enter their payment details and then later update them if they are incorrect.
T121	USR401	Customer should be able to enter their payment details so that they can place an order.	Passed	Customers can enter their payment details (i.e. paymentmethod, cardnumber, cvv, nameoncard).
T131	USR402	Customer should be able to save their payment details so that they can use them for future transactions.	Failed	The payment saving function is yet to be constructed.
T141	USR403	Customer should be able to update their personal (payment) details so that future transactions use the correct details.	In progress	Structure is provided to update personal payment details, however users cannot store these details as of yet.
T151	USR404	Customer should be able to have saved payment details entered into new orders so that they can save time placing subsequent orders.	Failed	The payment saving function is yet to be constructed.
T161	USR405	Customer should be able to receive an invoice so that as a guest they are able to track their purchases.	In progress	This aspect of the feature is out of scope, however the system still captures the ‘email’ address attribute from users, in order to perform this function in later iterations.
T171	USR502	Customer should be able to receive an order number so that they can track their order.	Passed	Customers receive and are presented with their order number when they begin their purchase order.
T181	USR506	Customer should be able to view their purchase history so that they can track their order(s).	Failed	The payment search function is yet to be set-up.

T191	(NEW)	Customer should be able to delete their saved payment detail before payment submission.	Passed	A customer can 'Cancel' their order before they select 'Pay'.

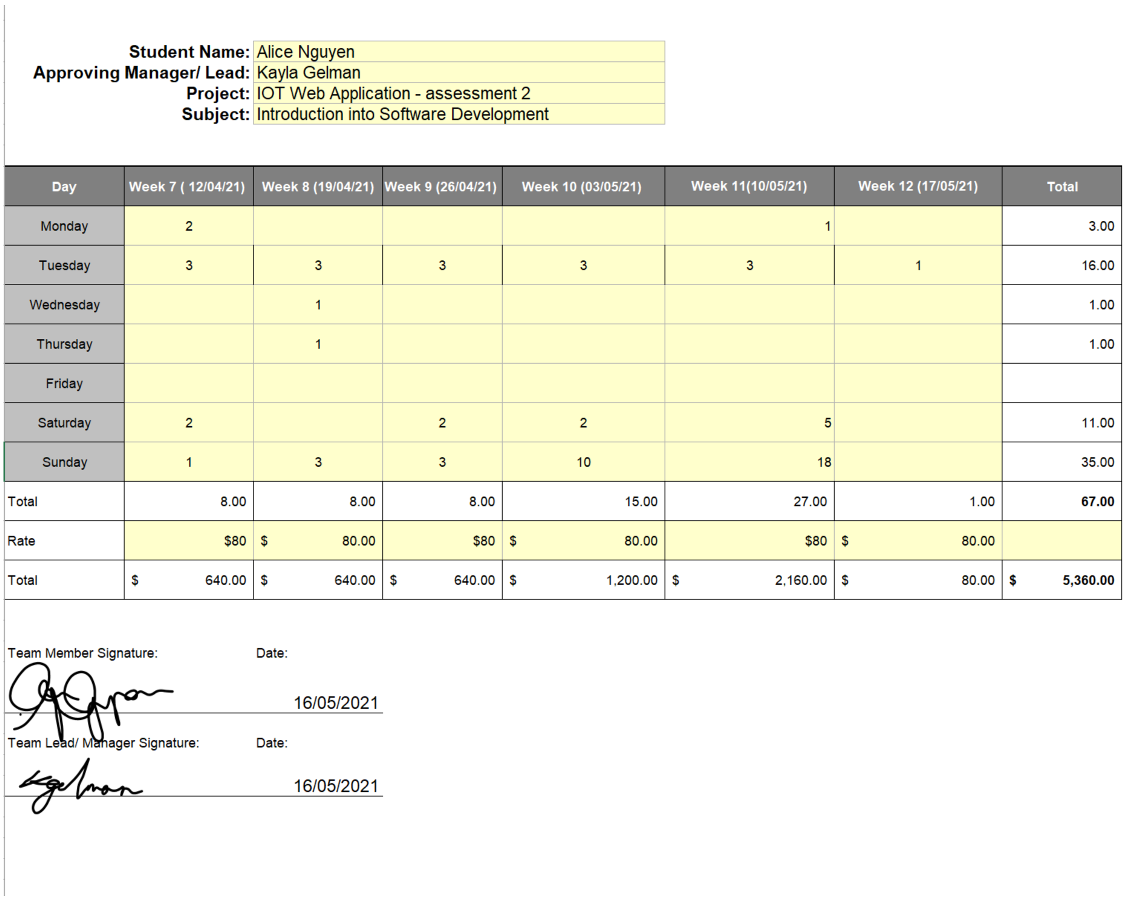
Defect Log:

DefectID	Description	Date	Test Case ID	Tester Name	Responsible	Status	Comments
C D1001	Error message when adding PaymentID column to table: "ALTER TABLE statement, the column has been specified as NOT NULL"	23/4/21	T111	Marina	Marina	Resolved	Added a 0 for 'default' field in column creation, which allowed non-null to be enabled. This was needed for the Primary Key (paymentID).
D1002	Error: incompatible types: java.lang.String cannot be converted to java.sql.Date	5/5/21	T121	Marina	Marina	Resolved	Change getCardExpiry getter method in PaymentModel java file from 'Date' data type to 'String', as specified in initialization.
D1003	EXCEPTION ERROR The number of values assigned is not the same as the number of specified or implied columns.	5/5/21	-	Marina	Marina	Resolved	Removed an extra comma in SQL query (for Payment table) which made it seem as though there were more values that needed to be populated in the 'insert' query.
D1004	TestPayment.java file output the text of two System.out.print() instances, rather than one, making it impossible to input data for just on input prompt.	5/5/21	-	Marina	Marina	Resolved	I was required to add 'in.next();' before 'paymentMethod = in.nextLine();' in order to break the input prompts with differing data types.

4. Appendices

4.1. Timesheets

4.1.1. Alice

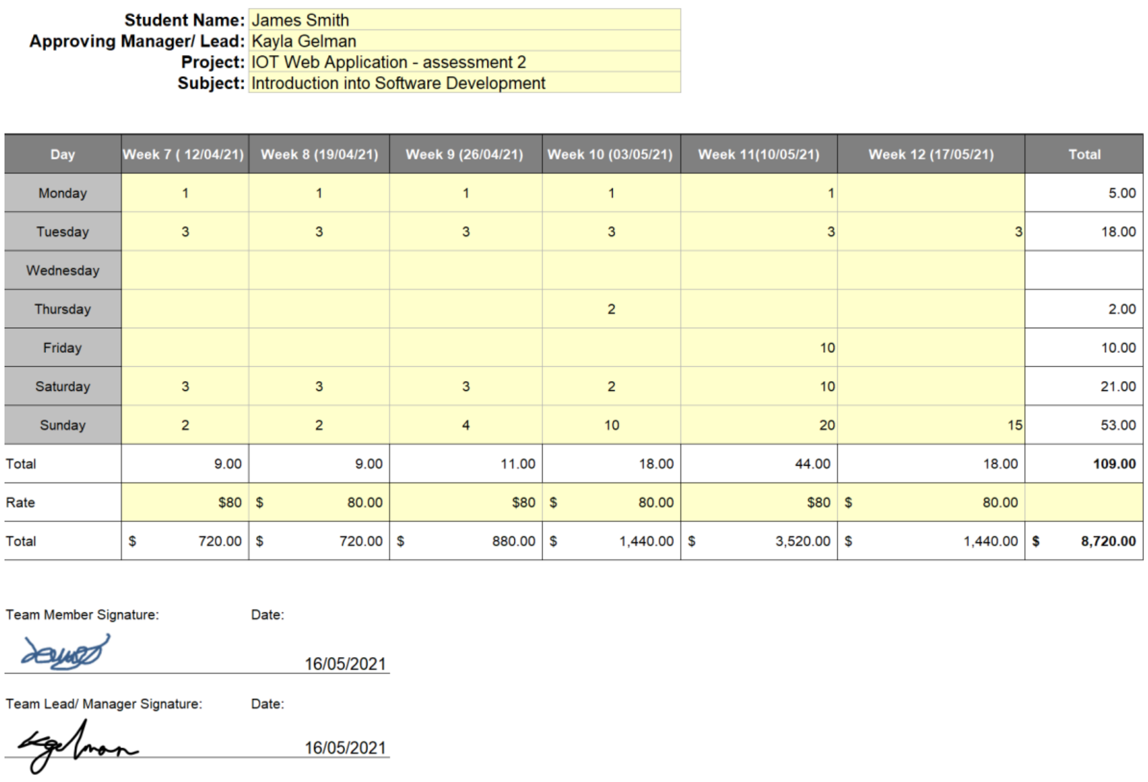


Student Name:	Alice Nguyen
Approving Manager/ Lead:	Kayla Gelman
Project:	IOT Web Application - assessment 2
Subject:	Introduction into Software Development

Day	Week 7 (12/04/21)	Week 8 (19/04/21)	Week 9 (26/04/21)	Week 10 (03/05/21)	Week 11(10/05/21)	Week 12 (17/05/21)	Total
Monday	2				1		3.00
Tuesday	3	3	3	3	3	1	16.00
Wednesday		1					1.00
Thursday		1					1.00
Friday							
Saturday	2		2	2	5		11.00
Sunday	1	3	3	10	18		35.00
Total	8.00	8.00	8.00	15.00	27.00	1.00	**67.00**
Rate	$80	$ 80.00	$80	$ 80.00	$80	$ 80.00	
Total	$ 640.00	$ 640.00	$ 640.00	$ 1,200.00	$ 2,160.00	$ 80.00	**$ 5,360.00**

Team Member Signature:	Date:
[illegible]	16/05/2021
Team Lead/ Manager Signature:	Date:
[illegible]	16/05/2021

4.1.2. James



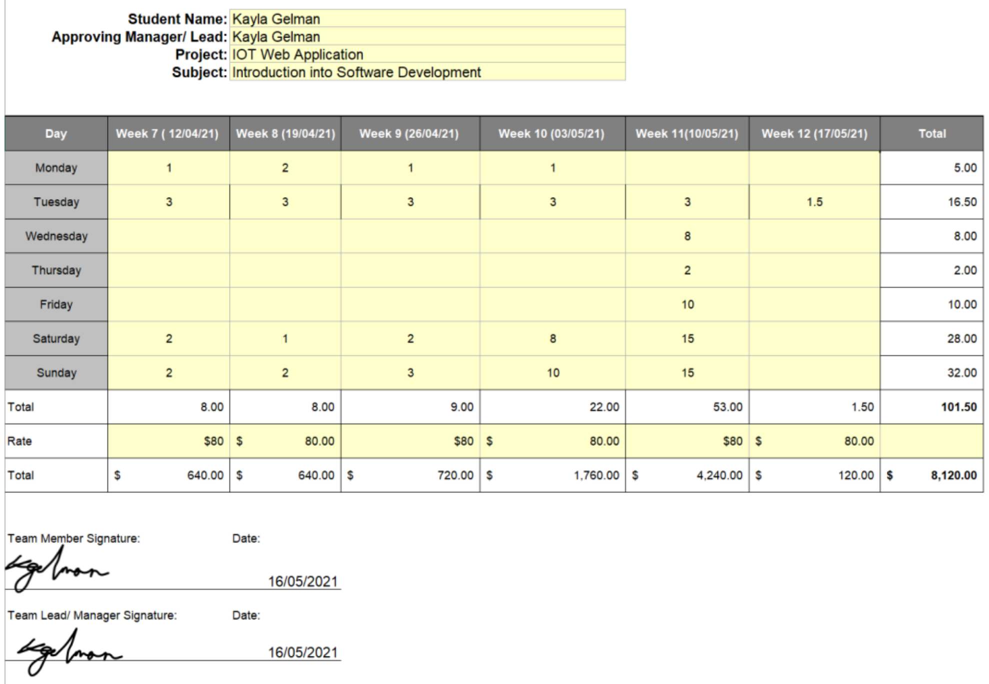
Student Name:	James Smith
Approving Manager/ Lead:	Kayla Gelman
Project:	IOT Web Application - assessment 2
Subject:	Introduction into Software Development

Day	Week 7 (12/04/21)	Week 8 (19/04/21)	Week 9 (26/04/21)	Week 10 (03/05/21)	Week 11(10/05/21)	Week 12 (17/05/21)	Total
Monday	1	1	1	1	1		5.00
Tuesday	3	3	3	3	3	3	18.00
Wednesday							
Thursday				2			2.00
Friday					10		10.00
Saturday	3	3	3	2	10		21.00
Sunday	2	2	4	10	20	15	53.00
Total	9.00	9.00	11.00	18.00	44.00	18.00	**109.00**
Rate	$80	$ 80.00	$80	$ 80.00	$80	$ 80.00	
Total	$ 720.00	$ 720.00	$ 880.00	$ 1,440.00	$ 3,520.00	$ 1,440.00	**$ 8,720.00**

Team Member Signature: [illegible] Date: 16/05/2021

Team Lead/ Manager Signature: [illegible] Date: 16/05/2021

4.1.3. Kayla



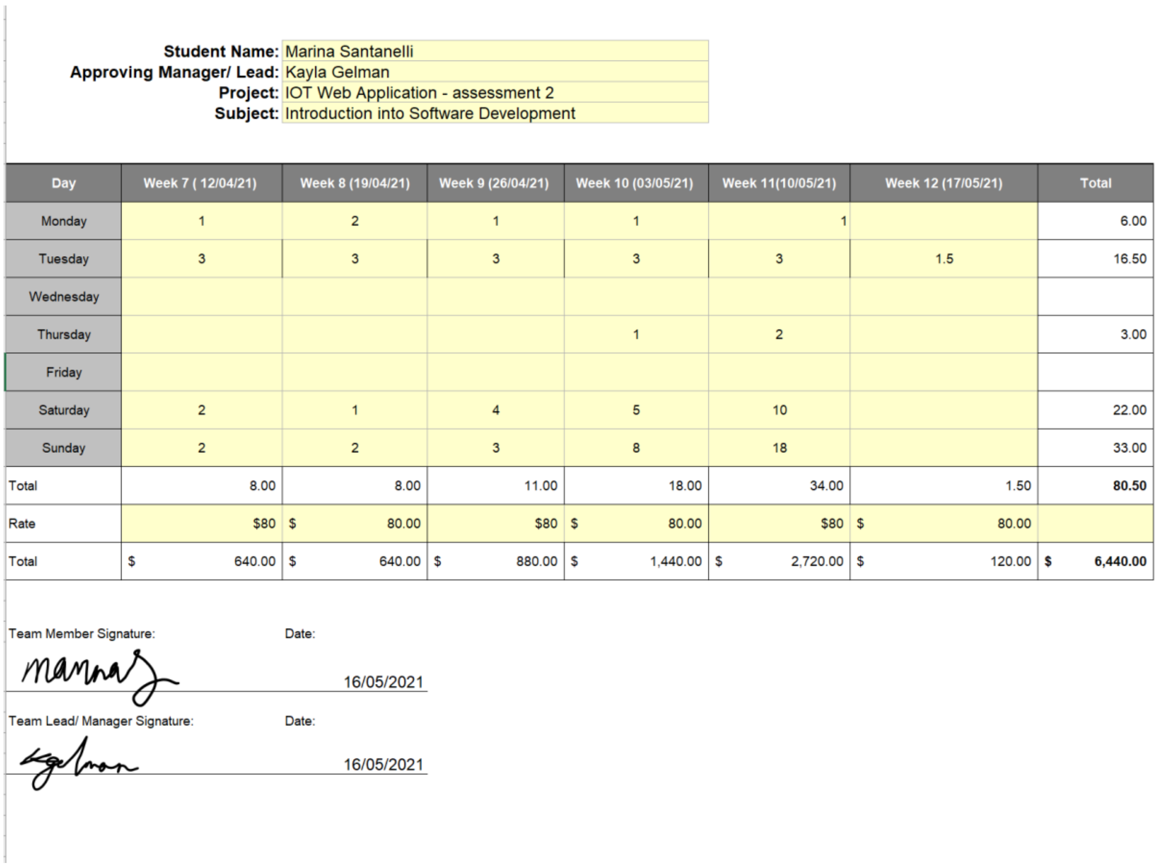
Student Name:	Kayla Gelman
Approving Manager/ Lead:	Kayla Gelman
Project:	IOT Web Application
Subject:	Introduction into Software Development

Day	Week 7 (12/04/21)	Week 8 (19/04/21)	Week 9 (26/04/21)	Week 10 (03/05/21)	Week 11(10/05/21)	Week 12 (17/05/21)	Total
Monday	1	2	1	1			5.00
Tuesday	3	3	3	3	3	1.5	16.50
Wednesday					8		8.00
Thursday					2		2.00
Friday					10		10.00
Saturday	2	1	2	8	15		28.00
Sunday	2	2	3	10	15		32.00
Total	8.00	8.00	9.00	22.00	53.00	1.50	**101.50**
Rate	$80	$ 80.00	$80	$ 80.00	$80	$ 80.00	
Total	$ 640.00	$ 640.00	$ 720.00	$ 1,760.00	$ 4,240.00	$ 120.00	**$ 8,120.00**

Team Member Signature: [illegible] Date: 16/05/2021

Team Lead/ Manager Signature: [illegible] Date: 16/05/2021

4.1.4. Marina



Student Name:	Marina Santanelli
Approving Manager/ Lead:	Kayla Gelman
Project:	IOT Web Application - assessment 2
Subject:	Introduction into Software Development

Day	Week 7 (12/04/21)	Week 8 (19/04/21)	Week 9 (26/04/21)	Week 10 (03/05/21)	Week 11(10/05/21)	Week 12 (17/05/21)	Total
Monday	1	2	1	1	1		6.00
Tuesday	3	3	3	3	3	1.5	16.50
Wednesday							
Thursday				1	2		3.00
Friday							
Saturday	2	1	4	5	10		22.00
Sunday	2	2	3	8	18		33.00
Total	8.00	8.00	11.00	18.00	34.00	1.50	**80.50**
Rate	$80	$ 80.00	$80	$ 80.00	$80	$ 80.00	
Total	$ 640.00	$ 640.00	$ 880.00	$ 1,440.00	$ 2,720.00	$ 120.00	**$ 6,440.00**

Team Member Signature: [illegible] Date: 16/05/2021

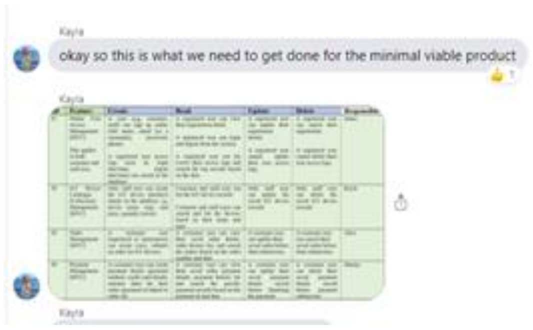
Team Lead/ Manager Signature: [illegible] Date: 16/05/2021

4.2. Individual Contribution Logbooks

4.2.1. Alice Nguyen- 12625582

Week 8: 20th April 2021

For this week's summary of contributions, I continued to use Facebook Messenger to communicate with my team. Team Leader Kayla revised a message that summarised what we needed to complete for the next iteration of the project, shown in the image below.

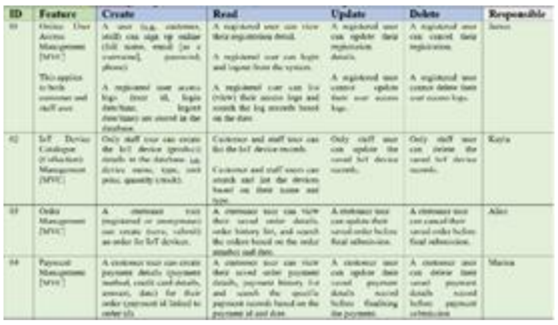


Kayla

okay so this is what we need to get done for the minimal viable product

Kayla

Kayla



ID	Feature	Create	Read	Update	Delete	Responsible
01	Online User Access Management [MVC] This applies to both customer and staff user.	A user (e.g. customer, staff) can sign up online (full name, email [as a username], password, phone). A registered user access logs (user id, login date/time, logout date/time) are stored in the database.	A registered user can view their registration detail. A registered user can login and logout from the system. A registered user can list (view) their access logs and search the log records based on the date.	A registered user can update their registration details. A registered user cannot update their user access logs.	A registered user can cancel their registration. A registered user cannot delete their user access logs.	James
02	IoT Device Catalogue (Collection) Management [MVC]	Only staff user can create the IoT device (product) details in the database. (e.g. device name, type, unit price, quantity (stock)).	Customer and staff user can list the IoT device records. Customer and staff users can search and list the devices based on their name and type.	Only staff user can update the saved IoT device records.	Only staff user can delete the saved IoT device records.	Kayla
03	Order Management [MVC]	A customer user (registered or anonymous) can create (save, submit) an order for IoT devices.	A customer user can view their saved order details, order history list, and search the orders based on the order number and date.	A customer user can update their saved order before final submission.	A customer user can cancel their saved order before final submission.	Alice
04	Payment Management [MVC]	A customer user can create payment details (payment method, credit card details, amount, date) for their order (payment id linked to order id).	A customer user can view their saved order payment details, payment history list and search the specific payment records based on the payment id and date.	A customer user can update their saved payment details saved before finalising the payment.	A customer user can delete their saved payment details saved before payment submission.	Marina

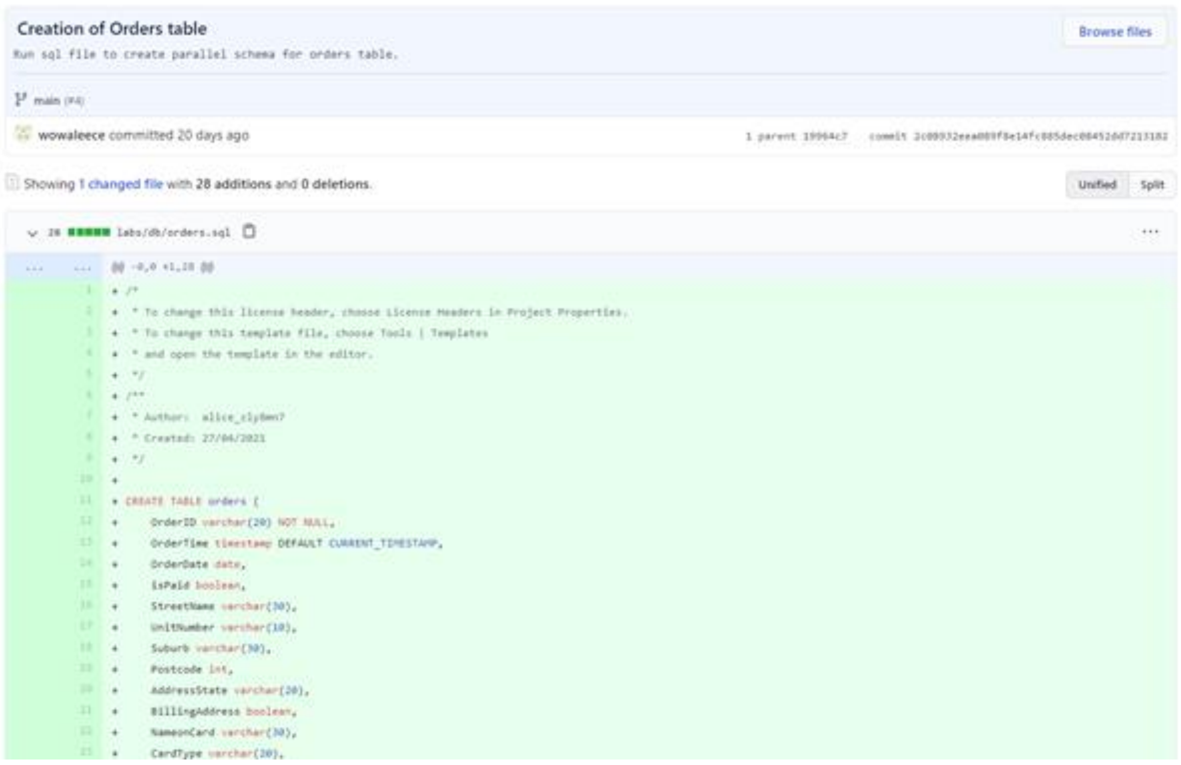
I nominated myself for Section 3, overseeing the CRUD operations for Order Management. In this week's workshop, I created a database and connected the program to that database to store data. The example provided in the workshop outline how the 'users' database could be created but this feature also applies to how the 'orders' database is created as well.

One thing that I was concerned over was the fact that the four of us were working on different features, all progressing at different times. This is concerning because upon merging the code, there may be merging errors because we would be using the database connection created on our local instances, rather than a single database. Therefore a thing I noted to discuss with my team would be creating the same database name, username, and password to prevent merging and pulling errors upon sharing code. Therefore over Messenger chat, we agreed to use the name 'shopDB' to avoid the future confusions.

Week 9: 27th April 2021

This week for me was planning about the approach I was going to take with my feature. When attending the workshop this week, we discussed about how elements of our project heavily overlap and are dependent on each other's implementation. This is both a good thing and a bad thing. Good thing in a way that we can practice our communication and building strong relationships as a team. However, it is a bad thing because those whose features rely on others are somewhat in limbo, unsure of where to start while they wait for others to implement. A solution to this was refer to all the planning that we did in Assessment Task 1. In building the MVC Model, Architecture Diagram and Data Diagrams we were able to establish a baseline of what variables and datatypes to use and how each class interacts with each other.

Since my feature was number 3, meaning mine was dependent somewhat on features 1 and 2, I was comfortable being able to write my features alongside my team members by referring to the planning documentation we submitted. Furthermore, I suggested that if any team member feels like they were sitting around waiting for someone else to complete their section that we focus on the UI of the system. This means that we could also complete some of the non-functional requirements listed in our user stories. The team and tutor seemed to like the idea and that was a plan for us to move forward.



Creation of Orders table

Run sql file to create parallel schema for orders table.

main (#4)

wowaleece committed 20 days ago   1 parent 19964c7   commit 2c09932eea089f8e14fc885dec08452dd7213182

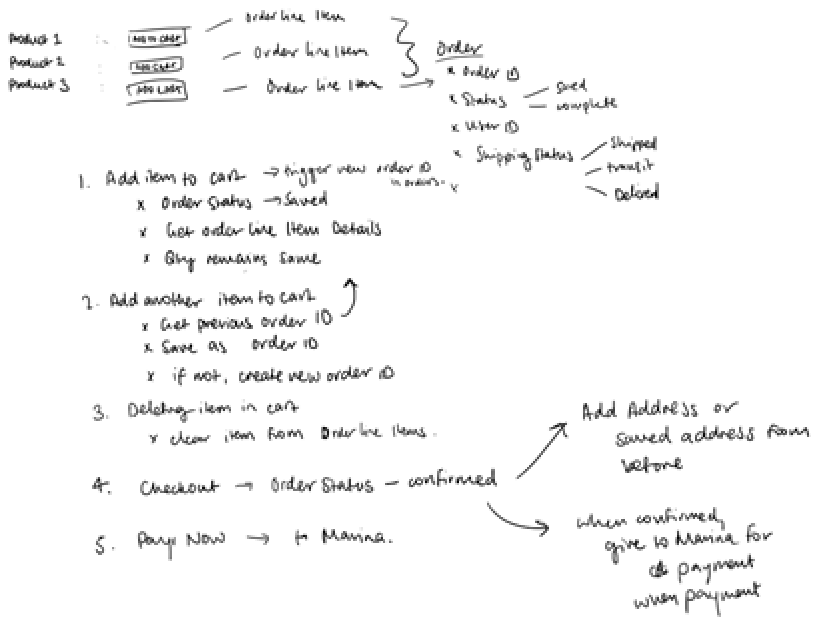
Showing 1 changed file with 28 additions and 0 deletions.

28 labs/db/orders.sql

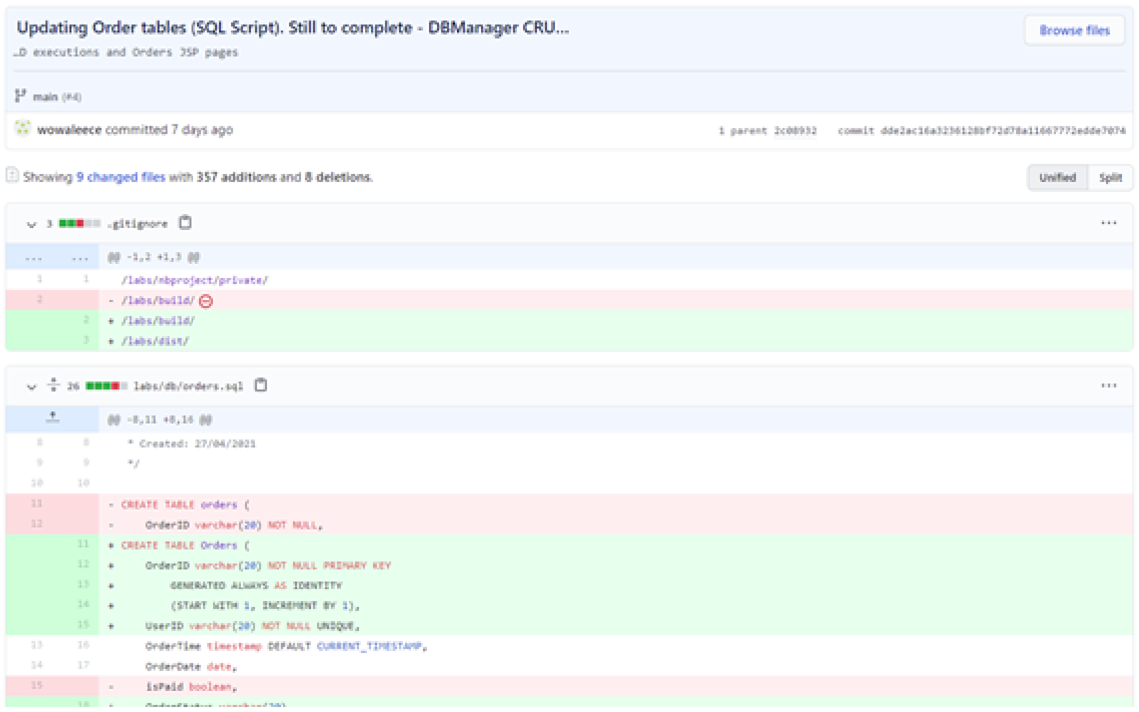
```
@@ -0,0 +1,28 @@
+ /*
+  * To change this license header, choose License Headers in Project Properties.
+  * To change this template file, choose Tools | Templates
+  * and open the template in the editor.
+  */
+ /**
+  * Author:  alice_clydem?
+  * Created: 27/04/2021
+  */
+
+ CREATE TABLE orders (
+     OrderID varchar(20) NOT NULL,
+     OrderTime timestamp DEFAULT CURRENT_TIMESTAMP,
+     OrderDate date,
+     isPaid boolean,
+     StreetName varchar(30),
+     UnitNumber varchar(10),
+     Suburb varchar(30),
+     Postcode int,
+     AddressState varchar(20),
+     BillingAddress boolean,
+     NameonCard varchar(30),
+     CardType varchar(20),
```

Our tutor also checked on our progress with our individual features and gave me some feedback on my table schema. Above is a screenshot of the git commit of the creation of orders table. Our tutor mentioned that ‘isPaid’ may be more useful as a String that stores ‘Status’ information rather than a Boolean that signifies if the order has been paid or not. We discussed that there is more than just 2 statuses an order can have, such as pending, incomplete, complete, saved and cancelled. Therefore, with this advice, I rewrote the schema to accommodate to the feedback.

Additionally, I used this week to sketch and mentally plan about different approaches to this project. Below is a rough scribble of notes on how I was going to implement the Orders feature.



Week 10: 4th May 2021



Updating Order tables (SQL Script). Still to complete - DBManager CRU…
…D executions and Orders JSP pages

main (#4)

wowaleece committed 7 days ago — 1 parent 2c08932 — commit dde2ac16a3236128bf72d78a11667772edde7074

Showing 9 changed files with 357 additions and 8 deletions.

```
3 .gitignore
@@ -1,2 +1,3 @@
  /labs/nbproject/private/
- /labs/build/
+ /labs/build/
+ /labs/dist/
```

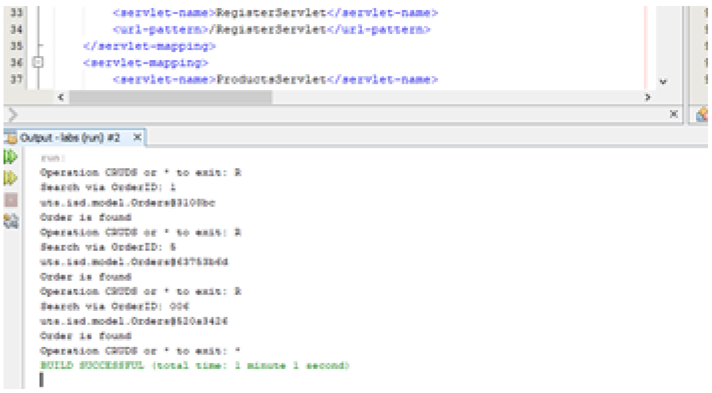
```
26 labs/db/orders.sql
@@ -8,11 +8,16 @@
   * Created: 27/04/2021
   */

- CREATE TABLE orders (
-     OrderID varchar(20) NOT NULL,
+ CREATE TABLE Orders (
+     OrderID varchar(20) NOT NULL PRIMARY KEY
+         GENERATED ALWAYS AS IDENTITY
+         (START WITH 1, INCREMENT BY 1),
+     UserID varchar(20) NOT NULL UNIQUE,
      OrderTime timestamp DEFAULT CURRENT_TIMESTAMP,
      OrderDate date,
-     isPaid boolean,
```

This week I finalised the Orders table and realised that I needed a separate one to store the line items within each order. Therefore OrderLineItems table was generated within this commit. I felt like I was falling behind this week, as I had yet to complete the workshop activities from last week. Therefore I spent this week catching up on Workshop 9 activities. Which was the implementation of the CRUD operations and queries to and from the database.

I found this quite difficult as I only worked with SQL briefly in past subjects. However, this was even further from my scope of understanding considering we were using Java prepared statements and result sets to query and communicate to and from the database. To take a break from the huge learning curve, I decided to create the base model files for Orders, including Orders.java and OrderLineItems.java. I also attempted to write the Views for the order features, including an add to cart view, view order history and view order feature.

Week 11: 11th May 2021

Things started to pick up this week as I finalised my CRUD operations from last week and began to implement controllers for Orders. I had thought last week's workshop was difficult, but this week proved to be the hardest week yet. Java Servlets and sessions became very confusing navigating between the M.V.C. elements of the project and I began to lose track of what each file contributed to the project. This was frustrating as I ran into multiple errors over again. However, one thing managed to work and that was my CRUD operations feature on my test controller. This is depicted below where it continuously finds order numbers from the database query.



```
        <servlet-name>RegisterServlet</servlet-name>
        <url-pattern>/RegisterServlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ProductsServlet</servlet-name>
```

```
run:
Operation CRUDS or * to exit: R
Search via OrderID: 1
uts.isd.model.Orders@3100bc
Order is found
Operation CRUDS or * to exit: R
Search via OrderID: 5
uts.isd.model.Orders@63753b4d
Order is found
Operation CRUDS or * to exit: R
Search via OrderID: 006
uts.isd.model.Orders@520a3426
Order is found
Operation CRUDS or * to exit: *
BUILD SUCCESSFUL (total time: 1 minute 1 second)
```

The team then started to communicate not only via Messenger chat but through voice channels such as Teams and Discord where we had peer to peer coding sessions. I believe this was good as problems I had were problems other team members experienced before which then saved us time to move onto solving the next problem that occurred. And with these open floor sessions, we were able to solve overlapping problems that would have costed us time if it were not brought up. I was able to implement the Orders and OrderLine Servlets however had a difficult time connecting the front view and the controller operations. This posed to be a common problem with the team as we all struggled to display the data. Below is a list of commits made for the assessment.

4.2.2. James

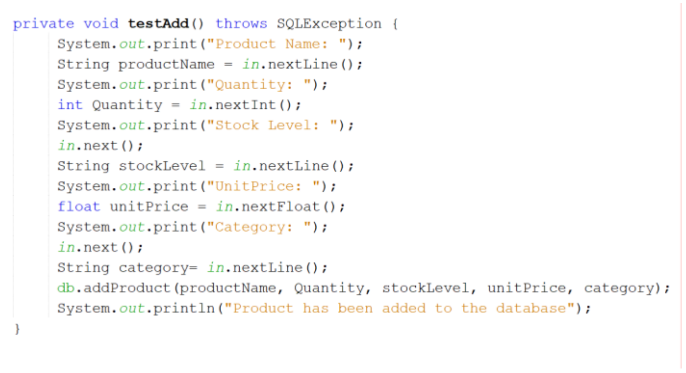
Week 8. 19/04/2021

This week, I'm focusing on getting a better understanding of the requirements for the project by designing the key pages needed to implement my feature, user management. More specifically, I focused on core login pages and learning about security concerns:

- Learned about how to properly handle login, attempted to learn about hashing functions.
- Learned major security concerns.
- Created Login queries for relevant CRUD commands (barring delete)

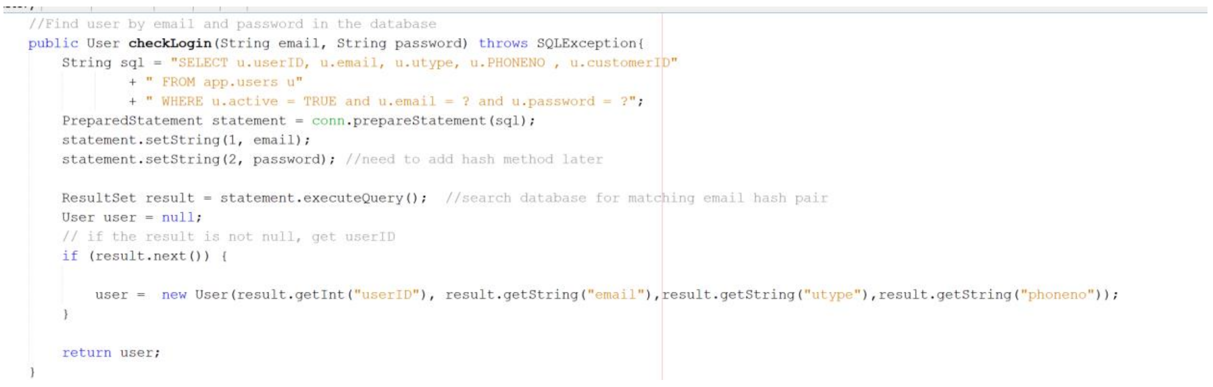
Week9. 26/04/2021

In this week after we learned how to use DAO for application logic, and I applied this to my Login functionality as best I could. I was hindered severely by not knowing about servelets. Unfortunately, I only learnt this upon reflection but I managed to troubleshoot Kayla's branch as she was having trouble executing SQL statements. To fix the issue she was having, I informed her that she would have to insert another line after executing nextInt() as this function will look out for an integer when it should be expecting a string.



```java
private void testAdd() throws SQLException {
    System.out.print("Product Name: ");
    String productName = in.nextLine();
    System.out.print("Quantity: ");
    int Quantity = in.nextInt();
    System.out.print("Stock Level: ");
    in.next();
    String stockLevel = in.nextLine();
    System.out.print("UnitPrice: ");
    float unitPrice = in.nextFloat();
    System.out.print("Category: ");
    in.next();
    String category= in.nextLine();
    db.addProduct(productName, Quantity, stockLevel, unitPrice, category);
    System.out.println("Product has been added to the database");
}
```

This week I also began implementing preparedStatements instead of executeStatements to secure against sql injection. An example can be seen below with my User checkLogin() function.



```java
//Find user by email and password in the database
public User checkLogin(String email, String password) throws SQLException{
    String sql = "SELECT u.userID, u.email, u.utype, u.PHONENO , u.customerID"
            + " FROM app.users u"
            + " WHERE u.active = TRUE and u.email = ? and u.password = ?";
    PreparedStatement statement = conn.prepareStatement(sql);
    statement.setString(1, email);
    statement.setString(2, password); //need to add hash method later

    ResultSet result = statement.executeQuery();  //search database for matching email hash pair
    User user = null;
    // if the result is not null, get userID
    if (result.next()) {

        user =  new User(result.getInt("userID"), result.getString("email"),result.getString("utype"),result.getString("phoneno"));
    }

    return user;
}
```

Week10. 03/05/2021

In this week we finally went in depth on servlets, enabling me to start developing out my login functionality to be used by the rest of the team. Unfortunately this was too little too late, and many of the basic login functionalities were being debugged and refactored until the last minute

Week11. 10/05/2021

This week I began investigating how to use Servlets more effectively, using doGet and doPost. I'm finding that I seem to be coming across some errors regarding when integrating the jsp page and Servlet. I also reworked the database design, at this point we were 1 week from release and I had not understood how to properly hash passwords so we decided to leave this functionality for later.

Week12. 17/05/2021

In the past week I attempted to refine my code and remove the many errors. The persistent errors currently are issues with flow control, where the request gets stuck in a servlet, or returns a bad value with no handling. These issues are mostly due to a knowledge constraint – as I only had 2 weeks to learn how to use Servlets, which have become the major source of errors in the application.

Given the opportunity I would have attempted to learn all the application logic, and JSP before week 8, as those things could be easily learned in a week or two. Meanwhile, having 4-5 weeks to learn and master servlets would have been advantageous, and allowed the time I needed to refactor my code as my understanding grew.

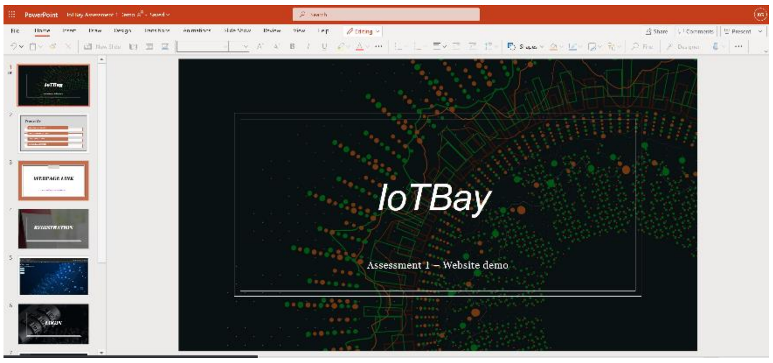
This week I also spent a lot of time debugging, mostly because it is difficult to debug servlets, and because I had to constantly bring the team up to speed with the current versions.

4.2.3. Kayla

Week 7 – Assignment 1 Showcase (12/04/21)

In this week's class, we showed our tutor our initial release for assessment 1. This experience was a lot less intimidating that I had initially anticipated. Initially, I thought that we were expected to present to the whole class and as a result, I prepared a whole PowerPoint explaining how we set up our initial NetBeans projects and worked as a group to pick our favourite view.

Unfortunately, this effort was not necessary as it was a very relaxed discussion on what we have completed, how our initial website is structured and where we plan to go from here.



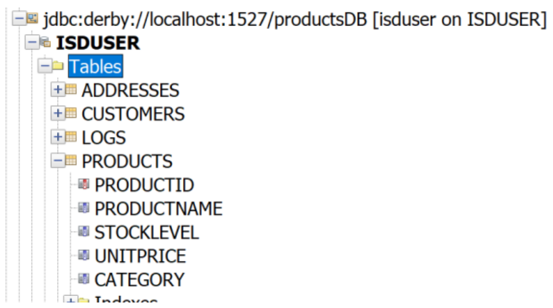
Week 8 – Application Datastore (19/04/2021)

In this week's workshop, we learnt about data persistence in MVC java web applications. We also assigned roles on who was going to work on what for the assignment, since Marian and Alice were away today, James and I assigned the first two functions to ourselves to complete.

During the workshop, we created tables in netbeans using my SQL. I was a little confused on how to create a table as the workshop looked like it used a manual way of filling in data. As a result, I found some other resources I could use to assist me in building my database:

https://netbeans.apache.org//kb/docs/javaee/ecommerce/setup-dev-environ.html

After following this tutorial, I created my initial ProductsDB. I did notice that our initial data dictionary was a little confusing to follow and upon reflection, planning out the connections as I try to code it makes me visualise the code a lot better.



In addition to setting up the database, we were introduced to running simple CRUD commands in SQL. I found this easy enough to understand as I've worked with SQL before. However, I'm still unsure of how I am going to set up the whole database whilst working in a team with other group members who might have different ways of thinking about how our database should be structured.

Week 9 – Application Logic (26/04/2021)

In this week's workshop, we looked at creating DAO modules to interact with he JDBC database. I was successful in setting up my database connection as per the activities we did last week and set up an initial DAO file. I have been struggling immensely to get my CRUD operations to work and am feeling incredible frustrated that simple operations are not connecting to the database.

I am still very confused on how I'm meant to print data in a jsp page and have been watching youtube video after youtube video. I tried to follow the online tutorial on canvas as best as I could but whilst the structure was the same for the DBManager file, I found that I kept running into compiling errors.

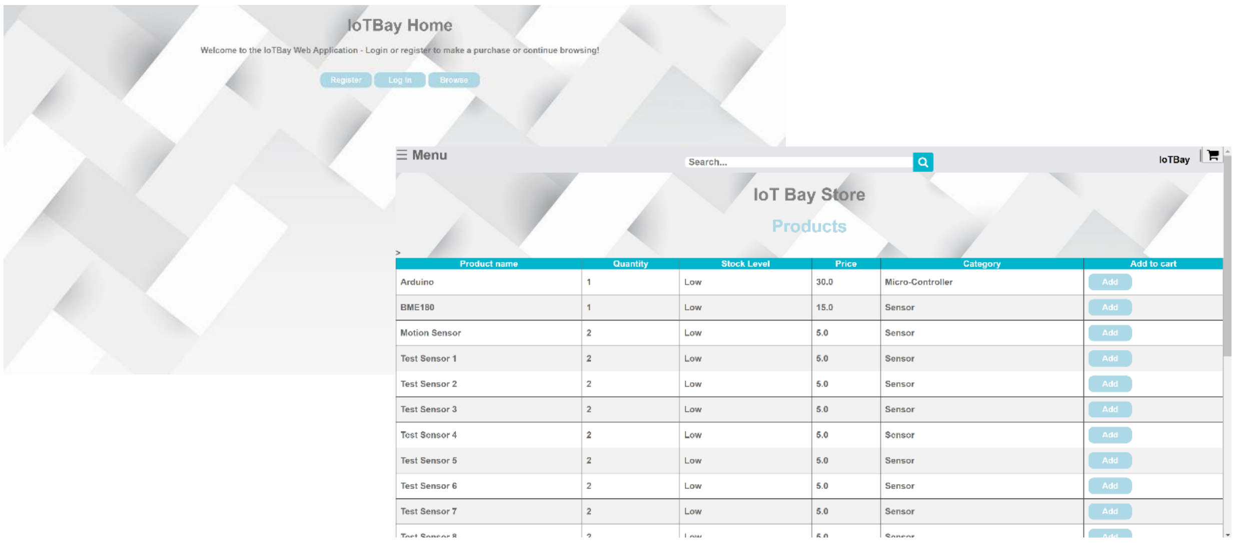
I am going to keep working through this but I am unsure on how to progress for the time being as I feel like if I don't get this working then I am going to get stuck cramming the assessment later down the line.

After many youtube videos, my CRUD commands are all finally working! I am over the moon happy that these commands are working as it means I can finally start creating my models and other views.

Week 10 – Putting it all together (03/05/2021)

I was unable to come into uni for the workshop this week, however, I began working on the servlets workshop at home. I wish we had been taught this sooner as it has all began slowly clicking how it all fits together. Whilst I'm happy I now know that's going on, it took hours of watching videos and debugging errors to figure out what I was doing wrong.

I also realised that the user interface wasn't too user friendly so I redesigned it:



IoTBay Home

Welcome to the IoTBay Web Application - Login or register to make a purchase or continue browsing!

IoT Bay Store

Products

Product name	Quantity	Stock Level	Price	Category	Add to cart
Arduino	1	Low	30.0	Micro-Controller	Add
BME180	1	Low	15.0	Sensor	Add
Motion Sensor	2	Low	5.0	Sensor	Add
Test Sensor 1	2	Low	5.0	Sensor	Add
Test Sensor 2	2	Low	5.0	Sensor	Add
Test Sensor 3	2	Low	5.0	Sensor	Add
Test Sensor 4	2	Low	5.0	Sensor	Add
Test Sensor 5	2	Low	5.0	Sensor	Add
Test Sensor 6	2	Low	5.0	Sensor	Add
Test Sensor 7	2	Low	5.0	Sensor	Add

Week 11 – Putting it all together – continued (10/05/2021)

This week was incredibly stressful, filled with days of debugging and figuring out why I was getting errors. We had an incredibly stressful time committing merges with our team mates and ensuring that our code was on the same track. There are some features I was sad to have only half implemented but due to the short timeline for this project, I found it hard to integrate all the servlets which was my groups main issue. Additionally, 2 hours before finalising our merges into one final repo, Marina lost her work as she accidentally merged an old brunch over her current changes. This was extremely devastating as we had spent all night making changes and finalising all of our servlets and functions.

4.2.4. Marina

Week 7 – (12/04/21)

Assignment 1 Showcase

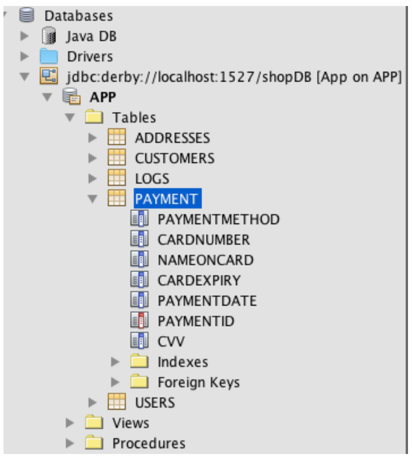
This week consisted of presenting our tutor with Assessment 1 (first release), which was the prototype of my group and I's IoTBay web application. At this point, we had merged our work together and created a consolidated prototype which had most of the necessary functions for the initial release.

Week 8 – (19/04/2021)

This week, I had personal priorities to attend hence I was away for the Tuesday tutorial lesson. However, I stayed in communication with my team members throughout the lesson, and they let me know that they had assigned me with the role of developing the feature 'Payment Management'. I was unsure what this entailed, so I looked at the assignment notification and became more familiar with what I was going to work on.

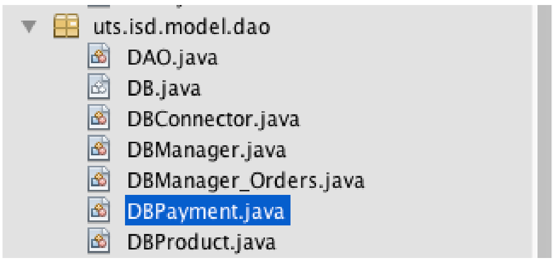
I caught up with this week's tutorial on Friday, where I completed the Lab exercise using SQL to create tables within a database. I practiced writing queries and became familiar with how to use SQL in NetBeans to insert data into the database and extract it, using the execute function.

I was then able to make a start on the database side of the assessment, where I implemented the 'PAYMENT' table in the database.



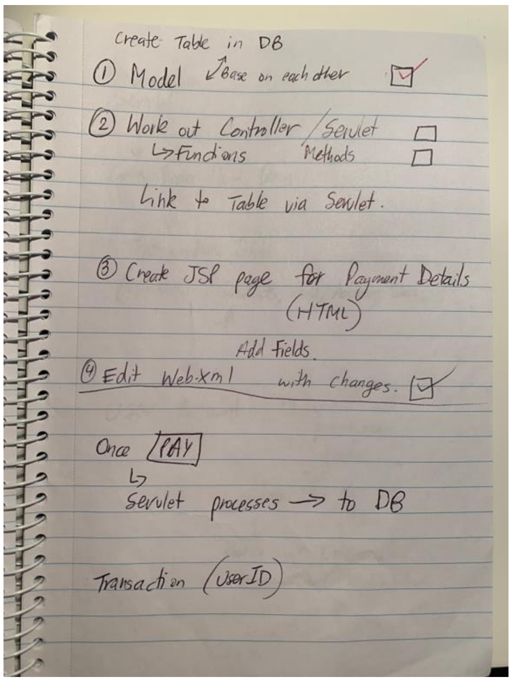
Week 9 – (26/04/2021)

This week I started to set-up the DBManager file, as per the online module on Canvas. I attempted this and then tried to create a similar file called DBPayment, to store methods relevant to my payment management feature.



uts.isd.model.dao
- DAO.java
- DB.java
- DBConnector.java
- DBManager.java
- DBManager_Orders.java
- DBPayment.java
- DBProduct.java

Other than this, I felt very overwhelmed with all the moving parts that are involved in the development of this web app. I was unsure of how to begin the controller and model sections mostly, as I was unclear how to integrate my ideas with the previous feature. It was hard for me to visualise what part I could start to develop when I didn't have the code from where I am supposed to begin from (i.e. order cart from Order Management feature). I discussed this with my group and they advised me to plan out the JSP pages, so that I could structure the view for the user and hence start to develop the UI of the web app. Furthermore, I created a rough plan of action as to what to approach first in my development.



Create Table in DB

① Model ↗↙ Base on each other ☑

② Work out Controller / Servlet ☐
↳ Functions Methods ☐

Link to Table via Servlet.

③ Create JSP page for Payment Details
(HTML)

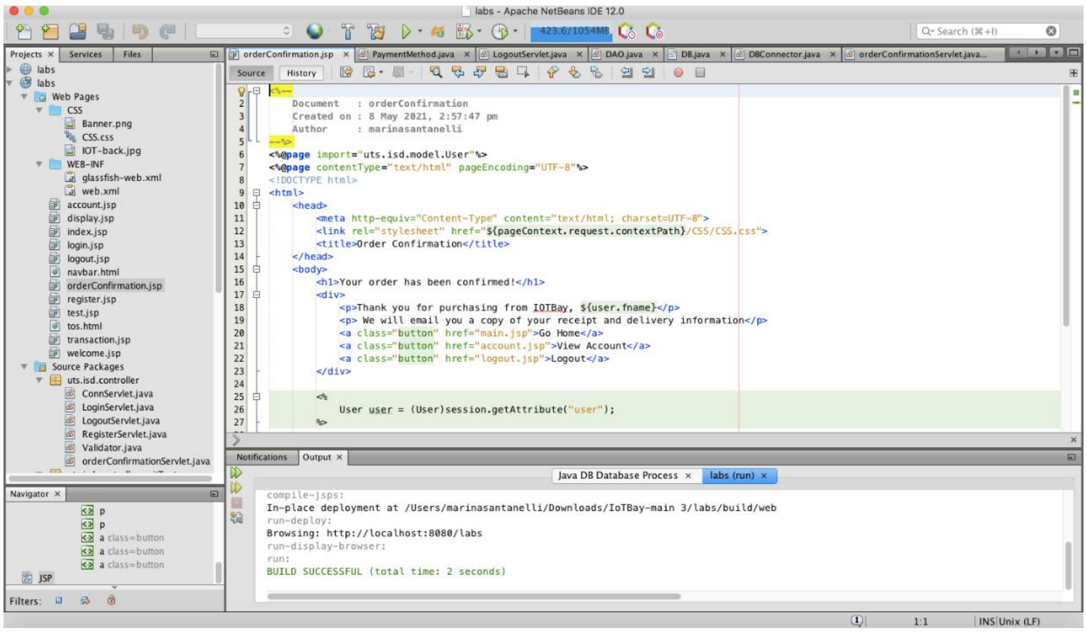
Add Fields.

④ Edit Web.xml with changes. ☑

Once $\boxed{\text{PAY}}$
↳
Servlet processes → to DB

Transaction (UserID)

- Example of work: set-up a draft of the 'orderConfirmation' JSP file for users to view when they have successfully completed their order transaction.

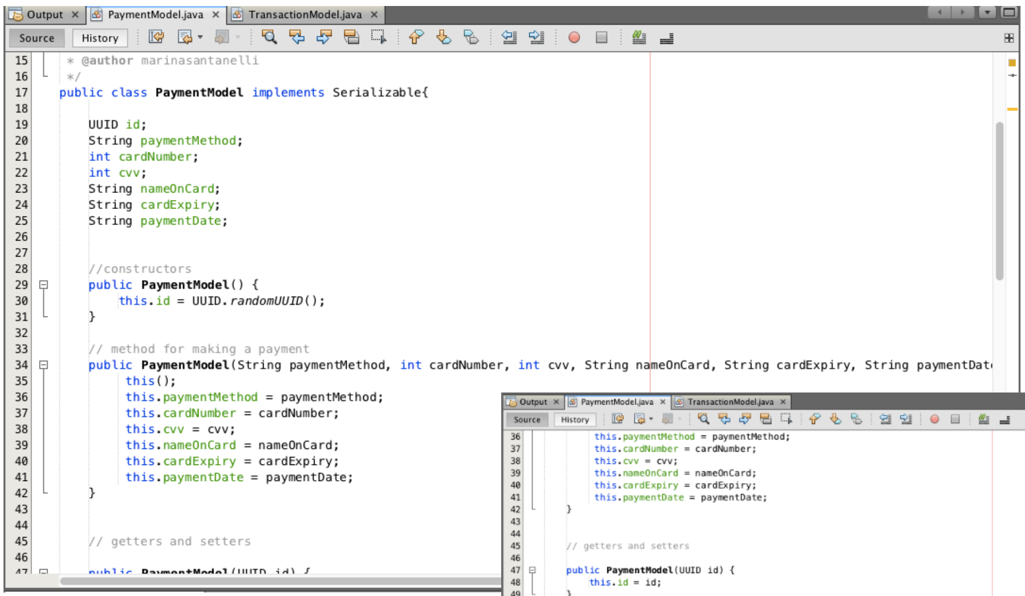


```
<%--
    Document   : orderConfirmation
    Created on : 8 May 2021, 2:57:47 pm
    Author     : marinasantanelli
--%>
<%@page import="uts.isd.model.User"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link rel="stylesheet" href="${pageContext.request.contextPath}/CSS/CSS.css">
        <title>Order Confirmation</title>
    </head>
    <body>
        <h1>Your order has been confirmed!</h1>
        <div>
            <p>Thank you for purchasing from IOTBay, ${user.fname}</p>
            <p> We will email you a copy of your receipt and delivery information</p>
            <a class="button" href="main.jsp">Go Home</a>
            <a class="button" href="account.jsp">View Account</a>
            <a class="button" href="logout.jsp">Logout</a>
        </div>

        <%
            User user = (User)session.getAttribute("user");
        %>
```

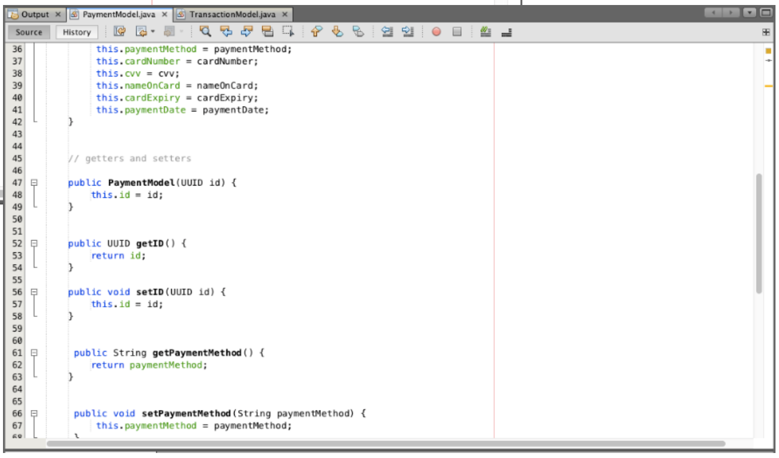
```
compile-jsps:
In-place deployment at /Users/marinasantanelli/Downloads/IoTBay-main 3/labs/build/web
run-deploy:
Browsing: http://localhost:8080/labs
run-display-browser:
run:
BUILD SUCCESSFUL (total time: 2 seconds)
```

Week 10 – (03/05/2021)

This week I started to develop the model for the payment management feature (i.e. PaymentModel.java), where I implemented getters and setters.

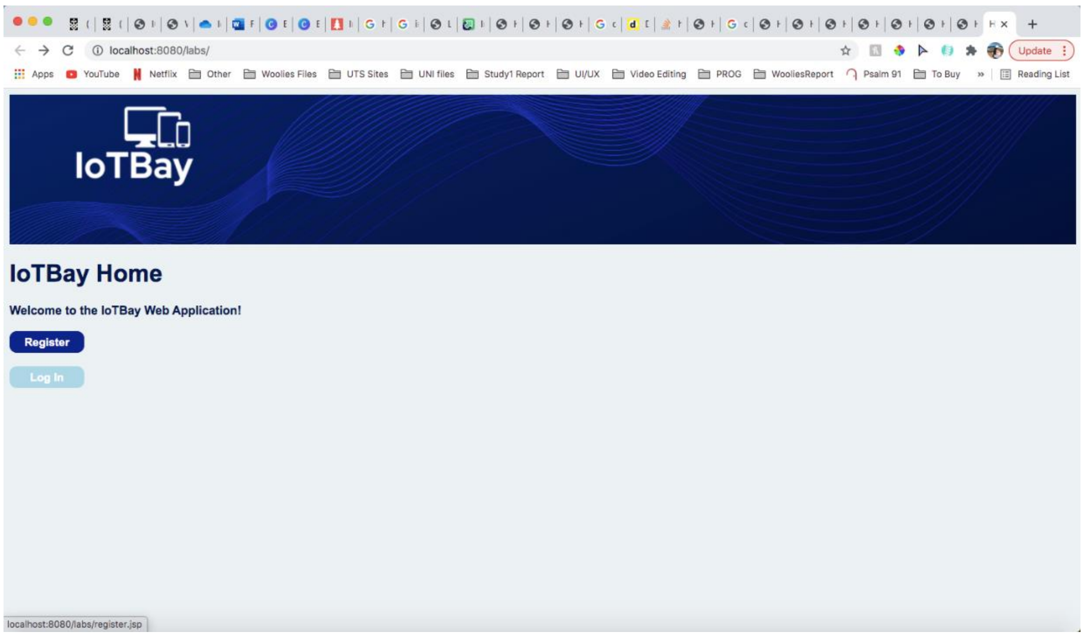


```
 * @author marinasantanelli
 */
public class PaymentModel implements Serializable{

    UUID id;
    String paymentMethod;
    int cardNumber;
    int cvv;
    String nameOnCard;
    String cardExpiry;
    String paymentDate;


    //constructors
    public PaymentModel() {
        this.id = UUID.randomUUID();
    }

    // method for making a payment
    public PaymentModel(String paymentMethod, int cardNumber, int cvv, String nameOnCard, String cardExpiry, String paymentDat
         this();
         this.paymentMethod = paymentMethod;
         this.cardNumber = cardNumber;
         this.cvv = cvv;
         this.nameOnCard = nameOnCard;
         this.cardExpiry = cardExpiry;
         this.paymentDate = paymentDate;
    }


    // getters and setters
```



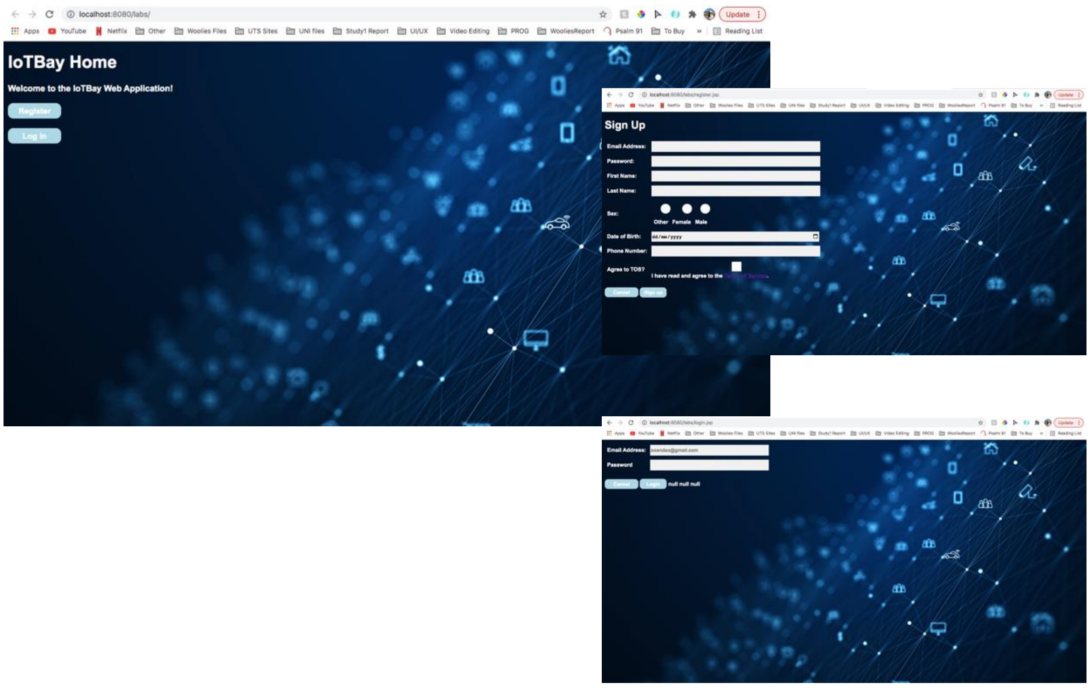
```
         this.paymentMethod = paymentMethod;
         this.cardNumber = cardNumber;
         this.cvv = cvv;
         this.nameOnCard = nameOnCard;
         this.cardExpiry = cardExpiry;
         this.paymentDate = paymentDate;
    }


    // getters and setters

    public PaymentModel(UUID id) {
        this.id = id;
    }


    public UUID getID() {
        return id;
    }

    public void setID(UUID id) {
        this.id = id;
    }


     public String getPaymentMethod() {
        return paymentMethod;
    }


     public void setPaymentMethod(String paymentMethod) {
         this.paymentMethod = paymentMethod;
```

I also attempted to redesign the UI, using the idea of a banner at the top of the page, as opposed to our plain and simple UI we had originally. I didn't like how it turned out (see below reason) but I got to experiment with CSS and I showed my group the ideas I had in regards to this design format.


localhost:8080/labs/
IoTBay
IoTBay Home
Welcome to the IoTBay Web Application!
Register
Log In
localhost:8080/labs/register.jsp

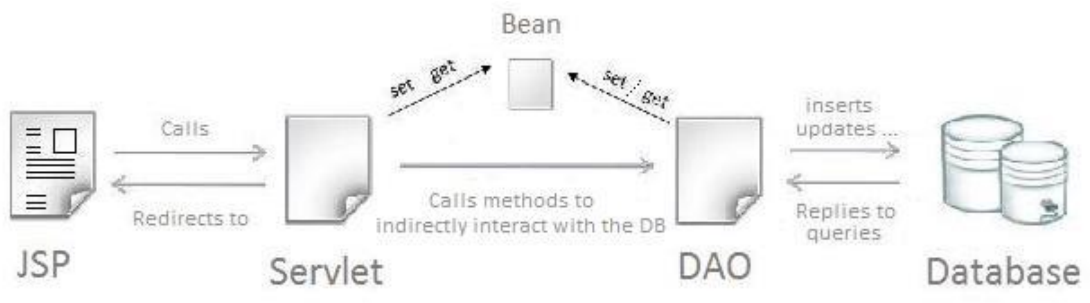
- Re-created a design for the interface but found that the white background colour did not allow the form fields where user's enter their registration details to be seen, as they were also white-coloured and thus blended in with the background.

This was in comparison to our older design:



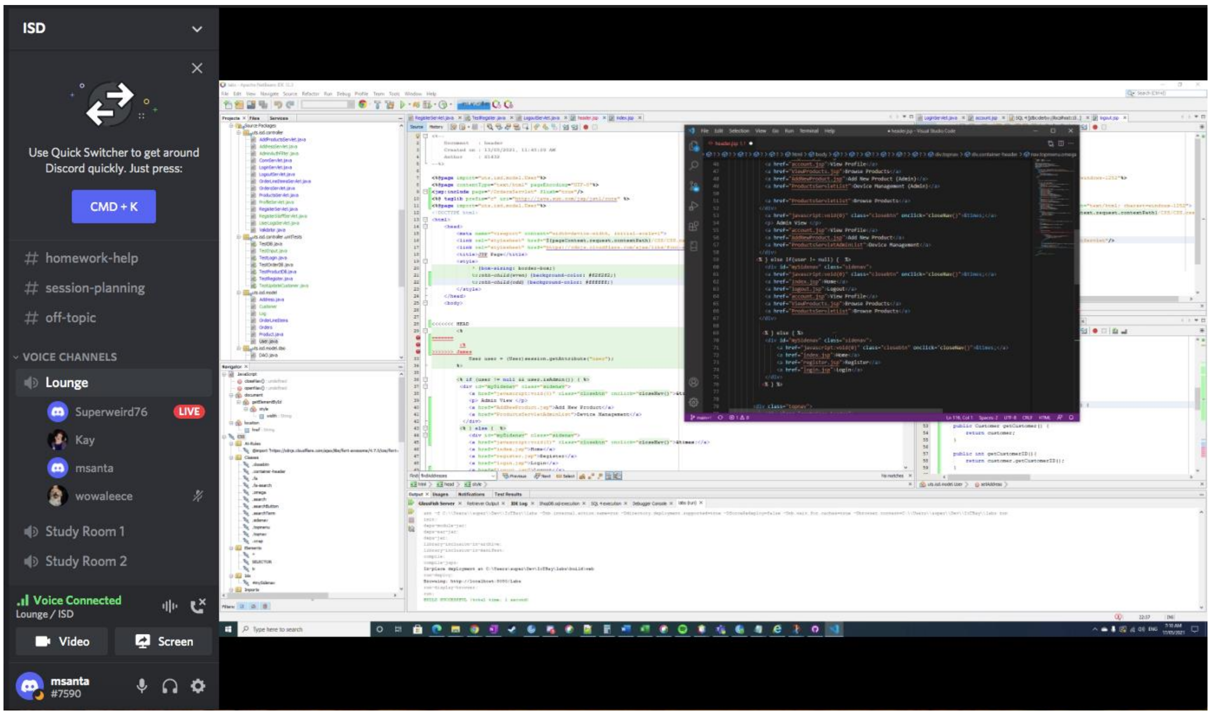
Week 11 – (10/05/2021)

This week involved adding the servlet pages into my project, which was very challenging for me as I could not understand how all the pages worked together to produce results. I investigated further into this and referred back to information in the Week 10 module of Canvas, including the provided diagram of the intended project structure.



I felt quite stressed this week as the due date was approaching rapidly, and I still had a lot of parts to complete and debug along the way. I also needed to consider the work of my team members, which became extra challenging for me as I didn't have clear insight into what the previous feature and its files looked like. However, I did communicate with my group about this and they helped me by sending me pictures and text messages of the information I required from them. We also had some commits in GitHub which I referenced to guide my development piece of the web app.

My team and I called each other frequently via Discord conversations and messaged via the Facebook Messenger app. During this time, we would help each other debug and solve complex issues that we could not resolve ourselves. We also took this time to reflect on what the final pieces were needed in order to complete the project to the best of our abilities.



4.3. Non-Functional Requirements (from Assessment 1)

Non-functional Requirements

Performance

User Story #	As a/an	I want to…	So that…
USR306	Customer	Receive a search result within 1 minute	finding a product is easier than reading the full product list
USR901	Customer/IoT Staff member	have a page returned within 1 minute of request	I am not frustrated while using the application

Security

User Story #	As a/an	I want to...	So that...
USR108	Customer/IoT Staff member	My Password to be hashed and salted	My password is not available through any SQL lookup.
USR405	Customer	Use HTTPS when making a payment	my payment details are encrypted in case of MIM attack.
USR107	Customer/IoT Staff member	My account's access to be restricted by a password	To keep my account secure.

4.4. Original Functional Requirements (from Assessment 1)

Project Backlog

User Story #	As a/an	I want to...	So that...	Estimates	Priority	Functional/Non-Functional	Story Type	Dependency	Status	Release	Iteration	Owner	Process/ Service
USR100 - Registration													
USR101	IoT Staff member	be able to register into the online application using my User ID (email) and password	I can access the system using credentials easily in the future	13	H	Functional	Story		In progress	R0	I3	Alice	admin view
USR102	IoT Staff member	be able to login to the online application using my registered User ID	I can manage items in the system	13	H	Functional	Story	USR101	In progress	R0	I3	Alice	Login (Staff)
USR103	IoT Staff member	logout of my account within 2 clicks*	I can log into another account easily if needed	8	H	Functional	Story	USR102	In progress	R0	I3	Alice	Logout (Staff)
USR104	Customer	Register into the online application using my User ID (email) and password	I can access the system using credentials easily in the future	13	H	Functional	Story		In progress	R0	I3	Alice	Register (Customer)
USR105	Customer	login to the online application using my registered User ID	I can browse and manage my orders and account	13	H	Functional	Story	USR104	In progress	R0	I3	Alice	Login (Customer)
USR106	Customer	logout of my account within 2 clicks*	I can log into another account easily if needed and to keep my account secure.	8	H	Functional	Story	USR105	To-Do	R0	I3	Alice	Logout (Customer)
USR107	Customer/IoT Staff member	My account's access to be restricted by a password	my account is kept secure.	21	H	Functional	Story		In progress	R1	I1	Alice	Login
USR108	Customer/IoT Staff member	My Password to be hashed and salted	My password is not available through any SQL lookup.	13	H	Non-functional	Story		To-Do	R0	I3	James	Login
USR110	IoT Staff member	Create users	I can moderate the system (Create, list, view, update, activate,deactivate users - staff and customers)	12	L	Functional	Story		To-Do	R1	I3	Kayla	Admin view (User Management)
USR111	IoT Staff member	Read users	I can moderate the system (Create, list, view, update, activate,deactivate users - staff and customers)	12	M	Functional	Story		To-Do	R1	I3	Kayla	Admin view (User Management)
USR112	IoT Staff member	Update user informatioon	I can moderate the system (Create, list, view, update, activate,deactivate users - staff and customers)	12	M	Functional	Story		To-Do	R1	I3	Kayla	Admin view (User Management)
USR113	IoT Staff member	Delete users	I can moderate the system (Create, list, view, update, activate,deactivate users - staff and customers)	12	M	Functional	Story		To-Do	R1	I3	Kayla	Admin view (User Management)
USR114	IoT Staff member	View application access logs	Moderate the system	8	M	Functional	Story	USR105	To-Do	R0	I3	Kayla	Application access logs
USR115	Iot Staff member	Application access logged	I can review logs at a later time	8	M	Functional	Story	USR105	To-Do	R0	I3	Kayla	Application access logs
USR200 - Change Details													
USR201	IoT Staff member	Change login details	I can still access my account if I get logged out	13	H	Functional	Story	USR101	To-Do	R0	I3	Marina	Update login details (Staff)
USR202	Customer	Change login password	I can still access my account if I get logged out	13	H	Functional	Story	USR104	To-Do	R0	I3	Marina	Update login details (Customer)
USR203	Customer	Change payment details	I can make sure my bank account details are up to date	8	H	Functional	Story	USR104	To-Do	R1	I2	Marina	Change payment details
USR204	Customer	Change email address	I can make sure confirmation purchase emails are sent to the correct account	8	H	Functional	Story	USR104	To-Do	R1	I2	Marina	Update Details

USR300 - Order Device													
USR301	Customer	Access the online application anonymously	I can browse products and purchase items while without comprising my privacy	21	H	Functional	Epic		To-Do	R0	I3	Marina	viewing application
USR302	Customer	be able to view IoTBay products online at any time	I can purchase at my own convenience	21		Functional	Epic		To-Do			Marina	viewing application
USR303	Customer	Browse IoTBay products	Compare similar products by price and specification	8	H	Functional	Story		To-Do	R1	I1	Marina	viewing application
USR304	Customer	be able to type in a search bar	I can find a product	13	H	Functional	Story	USR303	To-Do	R1	I2	Marina	viewing application
USR305	Customer	Add products to my "shopping cart"	Order multiple items together	8	H	Functional	Story	USR303	To-Do	R1	I1	Marina	viewing application
USR306	Customer	Receive a search result within 1 minute	finding a product is easier than reading the full product list	13	L	Non-functional	Story	USR304	To-Do	R1	I2	Marina	viewing application
USR400 - Payment													
USR401	Customer	Enter my payment details	I can place an order	8	H	Functional	Story	USR104	To-Do	R1	I1	James	Payment process
USR402	Customer	Save my payment details	I can use them for future transactions	8	H	Functional	Story	USR401	To-Do	R1	I2	James	Payment process
USR403	Customer	Update my personal details(payment details)	future transactions can use the correct details	8	M	Functional	Story	USR402	To-Do	R1	I3	James	Payment process
USR404	Customer	Have saved payment details entered into new orders	I can save time placing subsequent orders	5	M	Functional	Story	USR402	To-Do	R1	I3	James	Payment process
USR405	Customer	Receive an invoice	As a guest, I am able to track my purchases	21	M	Functional	Story		To-Do	R1	I3	James	Payment process
USR406	Customer	Use HTTPS when making a payment	my payment details are encrypted in case of MIM attack.	13	H	Non-functional	Story	USR401	To-Do	R1	I1	James	Payment process
USR500 - Manage Order													
USR501	Customer	Receive an email notification once I have placed an order	I know my order has been successful	8	M	Functional	Story	USR401	To-Do	R2	I2	Kayla	Payment confirmation
USR502	Customer	Receive an order number	I can track my Orders/Deliveries	5	M	Functional	Story	USR401	To-Do	R1	I3	Kayla	Order Tracking
USR503	Customer	Track my order using my order number	I can track my Orders/Deliveries	13	M	Functional	Epic	USR502	To-Do	R1	I2	Kayla	Order Tracking
USR504	Customer	View Purchase History	I can track my Orders/Deliveries	8	L	Functional	Story	USR502	To-Do	R1	I3	Kayla	Order Tracking
USR505	Customer	Save shipment details	I can track my Orders/Deliveries	8	L	Functional	Story	USR502	To-Do	R1	I2	Kayla	Usability
USR506	Customer	Save Purchases (history)	Repurchase previous orders	8	L	Functional	Story	USR504	To-Do	R2	I2	Kayla	Usability
USR600 - Performance													
USR601	Customer/IoT Staff member	have a page returned within 10 seconds of request	I am not frustrated while using the application	21	L	Non-functional	Story		To-Do	R2	I2	Alice	Performance
USR700 - View Inevtory Status													
USR701	IoT Staff member	View the status of all orders (Ordered, Delivered etc.)	I have visibility of my stock	21	M	Functional	Epic		To-Do	R1	I1	Kayla	Admin view
USR702	IoT Staff member	Track inventory levels	I can ensure appropriate stock levels for customers	21	M	Functional	Epic		To-Do	R1	I3	Kayla	Track inventory
USR703	IoT Staff member	Restock Inventory / update inventory quantities	I can ensure appropriate stock levels for customers	5	L	Functional	Story		To-Do	R1	I3	Kayla	Track inventory
USR800 - Usability													
USR-NF-801	All users	Validation errors should provide the admin user with a notification/banner	It is clear when there is an error and I need to change something	1	H	Non-functional	Story		To-Do	R1	I2	Alice	Usability
USR-NF-802	All users	The user interface should provide a consistent styling scheme	I am not overwhelmed on where to look	5	H	Non-functional	Story		To-Do	R1	I2	Alice	Usability
USR-NF-803	All users	User experience interactions should be simple and clear	It is easy to navigate through the program	1	H	Non-functional	Story		To-Do	R1	I2	Alice	Usability
USR-NF-804	All users	Icons should be large and easily identifiable	So that I know what to click in order to naviagte through the program	1	H	Non-functional	Story		To-Do	R1	I2	Alice	Usability