UNIVERSITY OF TECHNOLOGY, SYDNEY
41025 Introduction to Software Development
Project – Implementation & Testing
Assignment 2

Due Date: **Showcase in Week 12 During Scheduled Workshop [In-Person]**
Submission Due by Sunday of Week 12

Submission: Each group should assume the role of a Software Start-up Company, and will submit the following two items:

- **Working Software:** The individually implemented (modules or features) working software application code will be integrated and collated in a single project including a single database. Submit the ISD project working software code files (source and executable) with a README file (how to deploy and run the software) as a single .zip file via the 41025 ISD Assignment 2 page on Canvas. **You must not make any changes after the submission due date, otherwise the late submission rules will be applied.**
- **Report:** Each group will submit a softcopy (MS Word File or PDF) of the group report, including individual contributions or sections, with names and student IDs listed. Each individual contribution should contain (1) a brief description of the assigned software application feature or module, (2) non-functional aspects, and (3) software testing results with executed tests and (4) defect log, (5) individual contribution logbook/timesheets.

Your software code and report submission files should be named according to the following pattern: **Your workshop activity number–group id.** For instance, if your Wrk1 activity number is 02 and group id is G1 then your submission file should be named as 02-G1. Each team needs to nominate a project lead who will submit Assignment 2 (software and report) on behalf of the team. You must check the Turnitin report and ensure that your work does not contain plagiarism. Final Turnitin reports can be used as evidence by the teaching staff when plagiarism is suspected in an assignment. **Do not allow anyone to copy your solution – this is considered misconduct; all miscreants will receive a 0, at best for the assignment and will be dealt with according to University rules**.

Marks: 50%

Word Limits: There is no word limit. No student will be advantaged or disadvantaged based on the number of words or pages. **Focus on quality, not on the number of pages or words.**

Method: The assignment will be completed in groups, preferably the same group as for Assignment 1. Group size should be limited to **4-6 students** (enrolment numbers and situation-specific circumstance will dictate the actual group sizes). Groups were formed for Assignment 1 (during Weeks 1-4) and for any reason(s) you want to change groups for Assignment 2, it is your responsibility to find and join another group that is willing to accept you. You cannot change groups once you have started Assignment 2 (Week 7 onwards). **This is a group assignment, however each student must implement (code) and test end-to-end (following the MVC architecture layers) a fully functional and defect-free software feature or module as their individual contribution.** If the overall software works but an individual's implemented feature or module does not work according to the requirements, then that individual will receive 0 for this assignment. You must respect other students in the same group, different groups and the teaching staff. **If you have any group issues, then you must inform your workshop tutor as soon as possible and well before (at least 3 weeks or earlier) the submission due date. Group issues reported on or after the assignment due date may not be considered. There will be zero tolerance for any academic and non-academic misconduct. See the "Important guidance" page on Canvas for details.**

Objectives: Subject objectives:
1. Investigate and solve software development problems with minimal supervision.
2. Determine and balance the competing goals of software development activities within their constraints
3. Plan and manage a software development task to create, modify or extend a software feature or function to completion within the task constraints.
4. Apply sound software engineering practices to successfully create, modify or extend a software feature or function.
5. Clearly Communicate software and task information to interested stakeholders

Type: Project

Groupwork: Group submission, individually assessed

Criteria: The assignment will be individually assessed based on the following criteria.

Criteria Items	Objectives	Weight
Working software application implementation and demonstration (35 Marks)	1,2,3,4	70%
Tests of the solution (10 Marks)	1,2,3,4	20%
Overall quality, presentation (5 Marks)	5	10%
Total	-	100%

There will be no group marks. This means the individual marks for Assignment 2 shall be based on individual contributions in all three criteria across all the layers of the MVC architecture. **Please also note that there will be no negotiation on a non-functioning feature**. An individual's mark for this group work assessment shall be computed as:
Individual Student Contribution & Mark = Working software application [assigned feature] (all the layers of MVC) + Tests of the solution [assigned feature] + Overall quality, presentation [assigned feature]

Task: You are required to develop a web application for IoTBay. The software should be developed using agile practices and follow the MVC architecture designed and planned in Assignment 1, with appropriate adjustments discussed with your tutor. This assessment task will require a team of 4-6 students to produce, submit and present a **group report** (comprises of individual contributions), and a **working software application** (comprises of individually implemented and tested features or modules) and **individual contribution logbooks/timesheets** for Release 1 (see **Minimum Viable Product section**) for Online IoTBay. Based on the plan, software requirements, architecture, and design (Assignment 1) for Release 1, each group shall submit a:

- **Working Software Application**: Each student in a group will implement and test the assigned (as agreed among the group members and approved by the tutor as the product owner) feature or module of the software application; and a
- **Report:** Each student in a group will provide a brief description of the assigned software feature or module, the non-functional aspects, and software testing results with the defect log.

The individual contributions or parts will be collated in a group deliverable for submission and assessment (group submission but individual assessment). The deliverables of this assessment task also include a compulsory oral/visual presentation (**no slides**) of the individually implemented feature, individual contribution logbooks and the source code during the scheduled showcase. **Any whole team or individual student who fails to appear and present in the showcase will receive 0 as a final mark.**

Each group shall explore different ways of ensuring quality outcomes through the agile development and testing approach. This can be supported through a set of software development and testing tools. Students must get feedback on their work-in-progress project from their tutors during the workshop sessions before the formal submission.

Note: It is recommended that the essential functionality of the assignment will be implemented using the web technologies and techniques taught in this subject (e.g. Java, JSP, JDBC, Java DB). **You may choose to use a different technology or framework if you prefer, but you must ensure that the final software meets the criteria outlined in "Assignment 2: Consolidated Working Software Deliverables & Report" to receive marks. You may also use a CSS framework, such as Bootstrap, for the user interface**.

Minimum Viable Product

You have already implemented the initial prototype in Assignment 1. It was the starting point of the web application and provided the options of login and register for users (without database connectivity). It should include pages and links to other features (see the **Key Features Table** below), with proper navigation between views, controllers, models, and database tables populated with sample data. **In consultation with your tutor, you should select features from the "Key Features Table" for the minimum viable product, update the backlog, assign the feature to individual team members, and update the user stories captured in Assignment 1**. **Features 01, 02, 03 and 04 are mandatory and foundational features for the minimum viable product (Release 1).** If you have a 4-people group, then you will include these mandatory features. If you have a 6-people group, you will select the mandatory 4 features plus any other 2 features of your choice to ensure every student individually implements 1 feature following the MVC. The work distribution may change based on special circumstances or **group size (<3),** in consultation with the tutor or the subject coordinator. **The following Key Features Table demonstrates how to systematically organise features (e.g. data capturing/data management to BI & reporting).**

IoTBay Key Features Table (General Description: CRUD)

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.	e.g. name of the team member for the end to end delivery of this feature (MVC).
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.	
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.	
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	
05	Shipment Management [MVC]	A customer user can create shipment details (e.g. shipment method, date and address) for their order (shipment id linked to order id).	A customer user can view their saved order shipment details, history list and search the specific shipments based on the shipment id and date.	A customer user can update their saved shipment details record before shipment record finalisation for their order.	A customer user can delete their saved shipment details record before shipment record finalisation for their order.	
06	User Management [MVC]	System admin can create users (both customer and staff type users). *System admin is a built-in user with any software app.	System admin can view the created user records, user list and search the users based on their full name and phone number.	System admin can update user details and activate/ deactivate their status.	System admin staff can delete users.	
07	Customer Information Management [MVC]	System admin can create customer records (name, email, type, address).	System admin can view the customer record, customer list and search the customer based on their name and type [company or individual].	System admin can update customer details and activate/ deactivate their status.	System admin can delete customers.	
08	Staff Information Management [MVC]	System admin can create staff records (name, email, position, address).	System admin can view the created staff record, staff list and search the staff based on their name and position [e.g. salesperson, manager].	System admin can update staff details and activate/ deactivate their status.	System admin staff can delete staff.	
09	Supplier Information Management [MVC]	System admin can create supplier records (contact name, company, email, address).	System admin can view the supplier record, supplier list and search the supplier based on their contact name and type [company].	System admin can update the supplier details and activate/ deactivate their status.	System admin can delete suppliers.	
10	Data Management	System admin can create new records (e.g. user,	System admin can view records while bulk importing	System admin can update records in	System admin can delete or export	

	[MVC]	customer, staff) using bulk import to the system (e.g. CSV).	to the system for checking the data quality and confirmation.	the system via bulk import.	records from the system in bulk.	
11	BI & Reporting [MVC]	System admin can create reports and dashboard as requested by the various users (e.g. daily sales report, stock report)	Users (e.g. customer, staff) can run and view reports and dashboard as appropriate to their role and needs.	Users can customise reports and dashboard as appropriate to their role and needs.	System admin can delete reports and dashboard as requested by the users.	

*** User id, customer id, staff, order id, shipment id etc. can be autogenerated in database.**

Implementation Guide:

1. Landing page (home page) is the starting point of the web application and should provide the options of login and register to users and links to other pages (e.g. Device Catalogue, Order, etc). Allow users to logout or go back to your landing page from any other page. Follow general web application navigation mechanisms.
2. If a user cancels an order, then the order status is set to "cancelled". However, the cancelled orders are still stored in database.
3. If a user cancels their user account, all saved orders made by this user should be automatically cancelled and order details should be saved in the database with their status marked as "cancelled".
4. Once a user places an order, the number of devices should decrease accordingly. If a user cancels the order, the number of devices should be added back accordingly in the database.
5. If a device has 0 stock/quantity, users should not be able to order it.
6. As you add more pages to your web application, make them available from the home page or a relevant page.
7. A team should create a single database and populate the tables with at least 20 sample records each, such as users, devices, orders, etc. You can add the sample records through the database management system interface. This enables the development and testing of the features in parallel. For instance, while one student works on the order feature, he/she can use the product id and relevant details from the database without waiting for another student to complete the device catalogue management feature.
8. Your web application should perform validation of inputs to prevent system crashes. You should display an appropriate error message if the user has inputted incorrect data, allowing them to re-enter the data. The data validation should be **server-side not client-side** (**do not use client-side JavaScript or CSS for input data validation**). This is consistent with server-oriented web development principles. Verify the input data against corresponding data stored in the database, where applicable.
9. Your user interface should be well thought out, providing a consistent look and feel on all pages and useful navigation links. The user should be able to get to where he or she wants without ever having to click the browser's back button. You can use a CSS framework for the user interface.
10. Your code should be well designed, commented and neatly formatted.

Note: This project will challenge you from different perspectives and develop your professional capability and practical understanding including:

- "**Design is a code" (do you need upfront detailed plan and design? How much upfront plan and design is enough?**),
- **"Collaboratively developing/synchronising software code is challenging as a team"**,
- **"Adapting to change"** and
- "**What it takes to become a full-stack developer".**

Assignment 2: Consolidated Working Software Deliverables & Report

ITEMS	Maximum Marks	Note
Cover Sheet, Project Title/ Header Page & Individual Student Contribution Page	-	• **Cover Sheet & Project Header Page:** Sign, scan and embed the FEIT declaration of originality cover sheet containing the correct group name, student ids, names and signatures in the report just before the project title/header page. • **Individual Contribution:** At least 1 service/feature per team member covering all layers of MVC and the database for Release 1 (minimum viable product). Choose the features from for the "Key Features Table" **(see Minimum Viable Product Section)**. Include the individual contribution page with the following: Student ID, Name, Assigned Feature/Service. • **Roles:** Registered user (customer, staff), Anonymous user (non-registered customer), System admin (default built-in user for the application). **Note: If you do not include these then the assignment will not be marked, and you may receive 0 for the whole assignment.**
1. Working software application implementation and demonstration	**35**	**Working Software Code (MVC)**
View (CRUD) *Each student to develop the view for their assigned feature.	(10)	Working user interface (View) that: 1. Meet feature/user stories requirements (4 Marks) 2. Have a clear and consistent page layout, look and feel (1 Mark) 3. Contain navigation links between pages and the landing page (1 Mark) 4. Provide server-side validation of inputs (data) to prevent system crashes (2 Marks) 5. Provide an appropriate error message if the user has inputted incorrect data, allowing them to re-enter the data (2 Marks) **Note:** Do not use client-side JavaScript or CSS for data input validation.
Controller (CRUD) *Each student to	(10)	Working Controller(s) to control the data flow between the View and the Model, acting as an interface. A Controller receives requests from the

ITEMS	Maximum Marks	Note
develop the controller for their assigned feature.		View, processes them and performs server-side validations. The requests are further sent to the Model for data processing, and once they are processed, the data is sent back to update the View. The Controller should: 1. Meet feature/user stories requirements (8 Marks) 2. Contain Controller classes/pages/servlets to handle the data traffic between the Model (DAO) and the View (2 marks)
Model (CRUD) *Each student to develop the model for their assigned feature and connect it to the central database. No individual databases.	(15)	Working Model layer. This layer contains business logic of the system and represents the state of the application. It is independent of the View layer; the Controller fetches the data from the Model layer and sends it to the View layer. The Model layer including the connected Database should have following items. 1. Meet feature/user stories/data requirements (4 Marks) 2. Reusable Java Beans (4 Marks) 3. Data Access Object (DAO) & Database Connectivity (2 Marks) 4. Database Tables & Attributes (2 Marks) 5. Appropriate data constraints & relationships (2 Marks) 6. Sample data in the database – at least 20 records in each table (1 Mark)
2. Test of the Solution	**10**	**Report**
Software Application Feature *Each student to provide the description of their assigned feature.	(1)	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 1.
Non-functional Aspects *Each student to provide the description of their assigned feature.	(1)	Provide a brief description of the non-functional aspects of the assigned software application feature or module. Indicate any changes since Assignment 1.
Software Testing Results *Each student to	(4)	Document 1 acceptance test criteria and 1 Junit test for each user story relevant to your feature or service from Release 1. Record acceptance test case (linked to a user story), and Junit test case in the test matrix

ITEMS	Maximum Marks	Note
document acceptance test and testing results for their assigned feature.		(e.g. excel spread sheet or MS word table).
Defect Log *Each student to document defect log for their assigned feature.	(4)	Keep a log of the failed Junit test cases in a table or spreadsheet (each student must have their own test logs). The defect log should have the following items (you can include additional items): • Defect ID (DI001) • Defect Description (e.g. problem and action) • Defect Date • Test Case ID (e.g. Failed test case id) • Tester Name (e.g. who reported the defect) • Responsible (e.g. who will handle the defect) • Status (e.g. identified, assigned, in progress, resolved, unresolved defects) • Comments (any additional comments) • Summary: total defects, % of resolved defects, % of in progress defects.
Appendices – Project timesheet	-	Each student to complete and submit the timesheet signed by their project lead. **Note: Failure to submit the timesheet will result in a 0 mark for the assignment.**
Appendices – Individual Contribution Logbooks	**-**	Include contents from the Individual Contribution Logbooks. Link your individual contribution to weeks and hours recorded in timesheet. **Note: Failure to submit the individual logbook will result in a 0 mark for the assignment.**
3. Overall Quality, Presentation	**5**	Quality of visual/oral group presentation and software demonstration. You are not required to prepare and submit slides. Launch and present the report and the software from your laptop or as advised by your tutor. A mark breakdown is as follow: 1. Report is clear and easy to follow (2 Mark) 2. Software code is commented and neatly formatted (2 Mark) 3. Overall correctness of answers to questions (1 Mark)
Total	**50**	

Note: If the software fails to compile or run during the presentation or you do not present during the scheduled showcase, you will receive a 0 mark.

Assessment Feedback

Feedback on the marked assignments will be provided within 2 weeks after the submission due date. **You should regularly get feedback on the assignment tasks and deliverables from the tutors during the workshop sessions.**

Minimum Requirements

NO conceded passes are to be granted due to University Policy.

Referencing Standards

All material derived from other works must be acknowledged and referenced accordingly using the Harvard Referencing Style.

Late Penalty

See the "Important guidance" page on Canvas for late submission penalty.

Special Consideration

Special consideration for late submission, must be arranged beforehand with the subject co-ordinator (**email: yining.hu@uts.edu.au**). Please also see the UTS Special Consideration.

Accessibility Service

Students should email the subject coordinator as soon as possible (and prior to the assessment deadline) to make them aware of the impact on them meeting assessment component/requirements, and that they are seeking assistance through UTS Accessibility Service.

Academic Misconduct:

Please see the "Important guidance" page on Canvas for plagiarism and academic integrity in conjunction with the UTS Coursework Assessments Policy.

Querying Marks/Grades and Final Results

See UTS Coursework Assessments Policy for details.

Agile Values for Software Development

This subject would help you to reflect and understand the logic/ deep thinking underpinning the agile values.

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan