GROUP REPORT COMP3613

# ACA ENGINEERS MEMBERS:

Celine Ganar: 816008305

Aakil Ramlogan: 816007871

Anissa Harricharan: 816008114

**Software Engineering II**



**TABLE OF CONTENTS:**

**Introduction**……………………………………………………………………………….2

Problem…………………………………………………………………………………….2

Goals/Aims of the System………………………………………………………………….2

Target User…………………………………………………………………………………2

**Requirements and Analysis**………………………………………………………………3

User Requirements…………………………………………………………………………3

**System Requirements**………………………………………………………………..........4

Functional Requirements……………………………………………………………….......5

Non-Functional Requirements……………………………………………………………..6

**User Stories/Scenarios**………………………………………………………………........7

User Scenario 1: Logging In……………………………………………………………….7

User Scenario 2: Selecting a Destination and Searching for a Driver……………………..8

User Scenario 3: Being Picked Up………………………………………………………...9

User Scenario 4: Clicking on a Panic Button…………………………………………….10

User Scenario 5: Being Dropped off……………………………………………………..11

User Scenario 6: Rating a Driver………………………………………………………...12

**Requirements Justification and Increment Planning**………………………………...13

Requirements Solicitation and Justification……………………………………………..13

Iterative Requirements…………………………………………………………………..13

Scope of Increment………………………………………………………………………14

**Risk Analysis and Management**……………………………………………………….15

Requirements Validation………………………………………………………………...15

**Architecture**…………………………………………………………………………….16-17

**Design**……………………………………………………………………………………18

Components selected for Implementation……………………………………………….18

MakeABooking………………………………………………………………………….18

ActivatePanicButton…………………………………………………………………….18

Use Case Diagram……………………………………………………………………….19

Class Diagram for Overall System………………………………………………………20

Sequence Diagram……………………………………………………………………….21

**Proof of Methodology**…………………………………………………………………..22

Adopting SCRUM……………………………………………………………………….22

Testing Methodologies…………………………………………………………………..22-23

**SCRUM Methodology**………………………………………………………………….24

Sprint 1…………………………………………………………………………………..24-27

Sprint 2…………………………………………………………………………………..28-31

Sprint 3…………………………………………………………………………………..32-36

Sprint 4…………………………………………………………………………………..37-39

**Burndown Chart**…………………………………………………………………………40

**GitHub Repo Link**……………………………………………………………………….40

**Glossary**…………………………………………………………………………………41

**References**………………………………………………………………………………42

**INTRODUCTION**

***Project Name:*** UwiBus

***Problem:***

Students of the University of the West Indies (UWI) struggle daily with decent, efficient transportation needs to and from UWI as well as other destinations. With the current public transportation methods, it sometimes become troublesome for those students to reach UWI especially at rush hours. The arrival to classes in a timely manner becomes difficult as well as their safety is compromised. Since crime in Trinidad and Tobago is uncontrollable and UWI students often fall victim, there is an intense fear instilled within them. This becomes more concerning when traveling from their homes to UWI and vice versa, especially because of late classes scheduled. Additionally, ridesharing companies that exist may not feel completely safe or students cannot afford the high fare rates on a daily basis. Therefore, it is clear that a new, safe method of transportation is necessary for students and the implementation of UwiBus will assist effectively.

***Goals /Aims of System***

- Safe, efficient transportation to UWI and back home (GPS, Panic Button).

- Arrival to classes on time especially at rush hour.

- Affordable fares with daily specials as well as specials for loyal customers - Staff employment opportunities (Drivers).

***Target User***

The system is developed for UWI students in need of transport. Also, there is job opportunities for drivers.

**REQUIREMENTS AND ANALYSIS**

***User Requirements***

* The application will first prompt the user to create an account.
* The user should be able to choose not to enter payment information initially but will not be able to place drop orders.
* The tabs which will be present in the customer user interface are Account Details, Daily Deals, Map, Driver Rating and Panic Button and the driver user interface will have Account Details, My Orders, Rider Rating and Map.
* The application shall display the user’s current location on a map, along with an estimated time of arrival to destination.
* The user should be able to use the panic button if he feels threatened throughout the trip.

***System Requirements***

* The first time the application is opened, it must save the information entered by the user including a valid payment option.
* On the first use, the application will display a prompt asking the user to allow it to access his/her location.
* If a user requests a ride closest to him but that trip is full for that particular destination, the system will display text “Ride not available” when the driver is selected.
* When a device is not connected to the internet, the application will still display the user’s current location on the map on the screen but will ask that the user connect to the internet.

**SYSTEM REQUIREMENTS**

***Functional Requirements***

* The system shall allow the user to login using their credentials that was used when signing up.
* Users must be able to search for rides based on their location and intended destination.
* Users must be able to choose their preferred driver.
* They system shall allow the user to view the total cost of ride after booking.
* The system shall allow the user to cancel a ride.
* The system shall allow accurate GPS readings.
* Users must be able to pay when the ride is complete.
* Users shall be able to rate their experience and the driver once the trip is complete.
* The system shall allow the user to alert their emergency contact, campus police as well as the TTPS if the panic button is pressed, information will be sent to the named sources that a user is in distress as well as the location.

***Non-Functional Requirements***

**Performance/Efficiency**

* Keeps resource utilization as minimal as possible while providing relatively snappy response/update time (latency). This includes heavily managed memory allocation & maximum power savings while using GPS features.
* Maximum application start-up time of 5 seconds.
* Real-time GPS updates/refreshes at one second intervals.

**Scalability and Capacity**

* Ability to process multiple passenger/driver searches and allocation simultaneously without error/crashes.
* Able to host at least 10000 (expandable with app and user growth) user profiles in database system storage.

**Reliability**

* System accurately uses GPS map data to pinpoint and select destinations for travel.
* System also accurately calculates a reasonable cost for the trip and displays it to the user.

**Safety/Security**

* Passenger and driver location data can only be viewed by an administrator or an app monitor.
* Designated pick-up and drop-off locations are ensured to occur in safe, well-lit public areas.
* Passenger and driver’s basic profile information can only be viewed by each other or a system administrator providing no access to personal information by persons ineligible to use the application.
* Passenger’s payment methods and information are kept secure and only partially provided to system administrators for scenarios of error.

**Maintainability**

* Reliable updates of any changes in traffic rerouting/re-direction and geographical changes traffic flow.
* System administrators must ensure that user-tutorials and access to the application are provided to recently matriculated and registered students.

**Accessibility and Usability**

* UI easily understood by people ranging from ages 16 (UWI student/passenger) and above (drivers and administrators), with the ability to be easily navigated and utilized.
* Homogeneity between both Android and iOS applications, both applications provide the same easy-to-access UI and provides the same feature set.

**USER STORIES/SCENARIOS**

**Scenario 1: Logging In**

***Starting situation:*** The student or driver would like to use this application but first needs to authenticate their self by logging in.

***Normal flow of events:*** There is a prompt for a username and password when the application is opened. The user enters the relevant information and clicks on the login button to gain access.

***What can go wrong:***

* The user does not have an existing account for the application. This can be solved by creating an account by clicking on a hypertext link, “Sign Up”. The user will then be redirected to a page where he will enter the relevant information on a form to sign up.
* The user enters an incorrect username/password combination and clicks the login button which is immediately followed by a pop-up text stating, “Invalid password/username”. The user can rectify this problem by re-entering the correct username/password information. However, the user forgot his password. The solution for this problem includes a hypertext link, “Forgot Password”, which when clicked prompts the user for his registered email and the system emails him his username and password.

***Concurrent Activities:*** A user cannot be logged into multiple devices simultaneously else the previous logged on device will be logged out.

***System state on completion:*** The application will be opened with the user logged in. The user is redirected to a new menu screen where all the features of the application is available to him.

**Scenario 2: Make a booking**

***Starting situation***: The user (student) is logged in and would like to enter their desired destination and be matched with a driver close by for pickup.

***Normal flow of events:*** After successfully logging in, the user is prompted to enter their booking details: destination and the type of ride required. A map is displayed with the distances (in kilometers) of the user’s current location to the locations of the drivers in the area. The user accepts the available closest driver and information is displayed about the driver such as personal information or how long it would take to arrive at the passenger’s location.

***What can go wrong?***

* The user is still unsure how long the driver would take to arrive at their location as traffic and accidents are unpredictable. This can be solved by the user using the map tab to keep track of the driver’s location until time of arrival. The estimated time of arrival is also updated frequently while taking accidents and traffic into consideration.

***Concurrent Activities:*** While the driver is moving, the estimated distances are automatically calculated and instantly updated.

***System state on completion:*** From the list of drivers, the passenger will now see the driver of his choice and the relevant information.

**Scenario 3: Confirm Pick up**

***Starting situation:*** The user(student) is logged in and activates pick up.

***Normal flow of events:*** After logging on, the user can indicate that they have been picked up, this is necessary in order to use the panic button as well as calculating the cost of the ride since it is based on distance and how long the passenger is in the car.

***What can go wrong:***

* The user forgets to activate the pick up, this can be solved by using live location.

***Concurrent Activities:*** the driver also activates the pick up and the live location is monitored.

***System state on completion:*** The passenger can see the route being taken and where they currently are.

**Scenario 4: Activating Panic Button**

***Starting situation:*** The user (student) is logged in and is currently in route but feels unsafe and would like to raise an alarm.

***Normal flow of events:*** On logging in, the users may easily raise an alarm by clicking on the panic button which will send a message with their live location to an emergency contact and UWI estate police which will be forwarded the TTPS.

***What can go wrong:***

* The user accidentally clicks on the panic button. This can be solved by the user cancelling the panic button.

***Concurrent Activities:*** While the user is in distress, their live location is being shared with the relevant parties.

***System state on completion:*** The student will now see a range of options and measures they can take to protect themselves.

**Scenario 5: Confirm Drop Off**

***Starting Situation:*** The user(student) has arrived at his destination at the end of the trip and is prompted to give the driver a rating.

***Normal flow of events:*** After destination selection on the application and being picked up, the passenger is dropped off at either the University of the West Indies or a safe destination of his/her choosing at approximately the time calculated by the app. They then confirm on the app that they have been dropped off to the correct location.

***What can go wrong?***

* Time taken to arrive could be varied compared to the time calculated by the app as the app only provides an estimation of the time taken to arrive at an average speed of 80km/h. Other factors such as traffic peak time, road work and vehicle malfunction can affect travel time.
* The student could arrive at the wrong destination if they selected that destination whether by mistake or system malfunction.

***Concurrent Activities:*** The student leaves the vehicle choosing a suitable approval rating for their driver.

***System state on completion:*** The application thanks the passenger provided the trip and driver were successful and then restores itself to the home screen awaiting destination selection and prompts the user to add a rating.

***Scenario 6: Rating a Driver***

***Starting situation:*** The user (student) is logged in and would like to rate a driver after a ride.

***Normal flow of events***: On logging in, the user is able to see rides that they have completed and clicks on the one that they would like to rate via a pop up on the application. The user is able to rate (on a scale of 1-5) on safety, cleanliness and timeliness.

***What can go wrong:***

* The user chooses a rating that they did not intend to choose. This can be solved by allowing the user to alter the rating up to 24 hours after the initial rating.

***Concurrent Activities:*** The driver is also able to rate the passenger.

***System state on completion:*** The user will now see the details of their ride and the rating.

**REQUIREMENTS JUSTIFICATION AND INCREMENT PLANNING**

**Requirements chosen for core functionality:**

* The system shall allow the user/passenger to register for the application.
* The system shall be linked to a database.
* The system shall allow the user/passenger to search for rides based on his/her location and intended destination to be matched with the closest driver.
* The system shall allow the user/passenger to view the total cost of the trip as well as all the booking details after selecting the driver.
* The system shall allow the user/passenger to activate the panic button if he feels threatened throughout the trip.
* The system shall allow the user/passenger to leave a review for the driver.

**REQUIREMENTS SOLICITATION JUSTIFICATION**

The requirements selected constitute the core functionality of the proposed system, that is, the system can only be functional, unique and fulfil the idea with these included. These requirements are the registration and login profiles to ensure the safety of passengers and that user credentials are stored in case of emergencies. Also, without the ability to book the trip, the entire concept of the system is lost. Lastly, the panic button is also important because it provides a sense of safety and comfort to passengers if they are threatened during a trip. The review of a driver is essential as well because if given a poor review, further investigations may be made and the driver can possibly be fired.

**ITERATIVE REQUIREMENTS**

The sprints are as follows:

* **Sprint 1** – The system shall allow the user/passenger to register for the application AND The system shall be linked to a database.
* **Sprint 2** -  The system shall allow passengers to search for rides based on their location and intended destination AND allow passengers to view the total cost of trip together with all the booking details after booking.
* **Sprint 3** – The system shall allow passengers to activate the panic button if he/she feels threatened throughout the trip AND The system shall allow the passenger to leave a review of the driver.
* **Sprint 4** – Refactoring and Contingency.

**SCOPE OF INCREMENT**

The components of this project include design, implementation of code (java) and documentation. The team members will each use their skills to contribute towards building a piece of software which adapts to proper software engineering principles and SCRUM methodology. Extreme Programming Practices (XP) such as test-driven development (TDD) and refactoring would also be utilized in this project. Small releases which was also introduced in this project allowed developers to receive feedback and detect bugs at early stages. One way of doing so is continuous integration (CI) which was very important in the success of the application and following good software engineering principles. Lastly, the sprints would be delivered weekly after plannings and reviews at the beginning and ending of each sprint respectively.

**RISK MANAGEMENT**

*TABLE 1: Showing the Risks, Probability, Affects, Effects, Strategy Category and Strategies Employed to Handle These Risks.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk** | **Probability** | **Affects** | **Effect** | **Strategy Category** | **Strategies To Be Employed** |
| Requirements Change | Low | Project and Product | Serious | Risk Reduction | Design software to allow flexibility. |
| Project is Behind Schedule | Moderate | Project | Tolerable | Risk Reduction | Encourage the early completion of subsequent components. |
| Incorrect Budget Estimation | High | Business | Serious | Risk Reduction | Implement contingency plans for various stages of the project. |
| Staff ill at Critical Time | Moderate | Project and Product | Serious | Risk Avoidance | Employ workers to shadow other workers / Pair Programming. |
| Project Competition | Low | Project | Tolerable | Risk Reduction/ Contingency | Add functionalities that may be absent in competitors project. |
| Inaccurate Stakeholder Expectations | Low | Project and Product | Serious | Risk Avoidance | Provide meetings and documents to let them know the goals of the project. |
| Staff Demotivated | Low | Project and Product | Serious | Risk Reduction | Reward employee work |
| Lack of Profitability | Moderate | Business | Tolerable | Risk Reduction | Gain good publicity for project even before its completed. |
| User Rejection | Moderate | Product | Catastrophic | Risk Avoidance | Iterate new versions more to users’ likings/desires. |

**REQUIREMENTS VALIDATION**

The User Requirements were written in natural language called User Stories to allow for easy understanding by a non-technical audience. They were adjusted until they covered all required functionality for UWIBUS’s daily operations. The development team then selected the requirements that encompassed the core functionality for the increments.

**ARCHITECTURE**

We chose the Model-View-Presenter (MVP) architecture because it seemed most fitting for our application and design goals. Some advantages of this architecture that we took into consideration are:

1. Clear separation of responsibilities between components. This separation allows for an easier understanding and maintenance of the code base.
2. Modularity. Modularity allows you to, for example, switch to a different implementation of view component in order to completely change application's UI, while all other components remain intact.
3. Easier testing. Since there are well defined boundaries between components, it becomes much easier to test each component in isolation.

The MVP architecture separates functionality into the following components:

***Model***

A model encompasses business objects, and in the case of this application we have:

* Student - This represents a passenger which is a student and has basic information including student ID, student name, student number, emergency contact and email address.
* Driver - A representation of a driver’s basic information including driver ID, driver name, phone number and email address.
* Booking - booking information such as booking ID, student ID, time, date and destination.
* Trip - a representation of a trip ID, the driver’s ID, booking ID, the rating, time and location coordinates.
* EmergencyContact - details of student’s emergency contact, their contact’s name, number and email.
* PanicActivated - details of incidents such as panic ID, student ID, location, date, time, driver ID.

***View***

A view contains all the UI components that make up the application. This includes any tables, labels, buttons, textboxes, etc. Views are responsible for the layout of the UI components and have no notion of the model.

The views in the application are:

* LoginOrSignupView - Users would be able to login as a student or driver or if they are new instructions to create a new account will be prompted.
* BookingView - the passenger will be able to input their destination either manually or by dropping a pin on the map. They will then choose their preferred ride type and then select a driver who is closest to them.
* DriverDetailsView- once the student selects their driver, all the necessary information about that driver is displayed to them as well as the drivers live location.
* ConfirmPickupView - the passenger can verify that they have been picked up safely and the meter will now being so that the cost of the ride can be calculated.
* ActivatePanicButtonView - the user can activate the panic button by clicking on it.
* ConfirmDropOffView - the passenger confirms that they have been dropped of safely.
* RatingDriverView - the passenger can leave rating on a driver they previously had a trip with.

***Presenter***

A presenter contains all the logic for the application. As a general rule, for every view you’ll want a presenter to drive the view and handle events that are sourced from the UI widgets within the view.

For our application we have the following Presenters:

* LoginOrSignupPresenter - Checking the database for valid login credentials and validating fields during sign-up.
* BookingPresenter - Checking for available rides based on the information entered by the passenger.
* DriverDetailsPresenter - Displaying all available information on the selected driver.
* ConfirmPickupPresenter - Displaying information on ride, the live location and the calculated cost thus far.
* ActivatePanicButtonPresenter - Notifying the passenger that the panic button has been activated and help is on the way.
* ConfirmDropOffPresenter - Displaying the actual cost of the ride and other information such as duration.
* RatingDriverPresenter - Displaying scale between 1-5 to rate driver as well as view average rating of that driver.

**DESIGN**

**Components selected for implementation:**

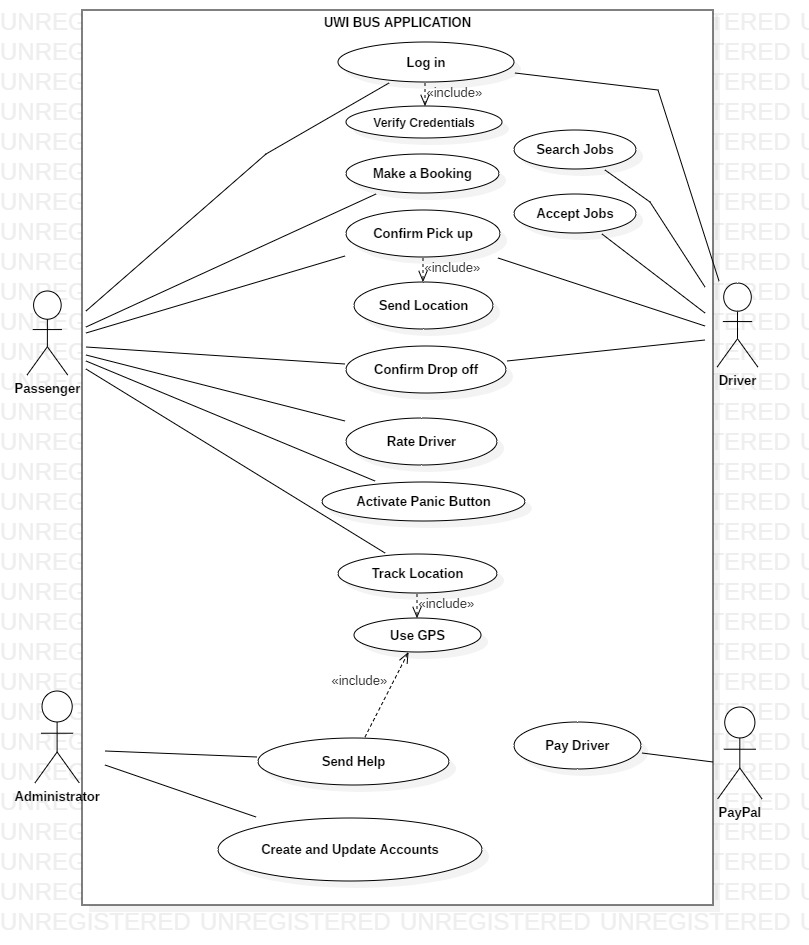
***MakeABooking***

* Models: Student, Driver, Trip
* View: BookingView (Map Widget, Destination textbook, Confirm button, DriverDetails, Ride Type checkbox, Driver selection checkbox)
* BookingPresenter (int bookingID, int studentID, string Time, Date date string destination, getStudentName(), CreateBooking(), CancelBooking())

***ActivatePanicButton***

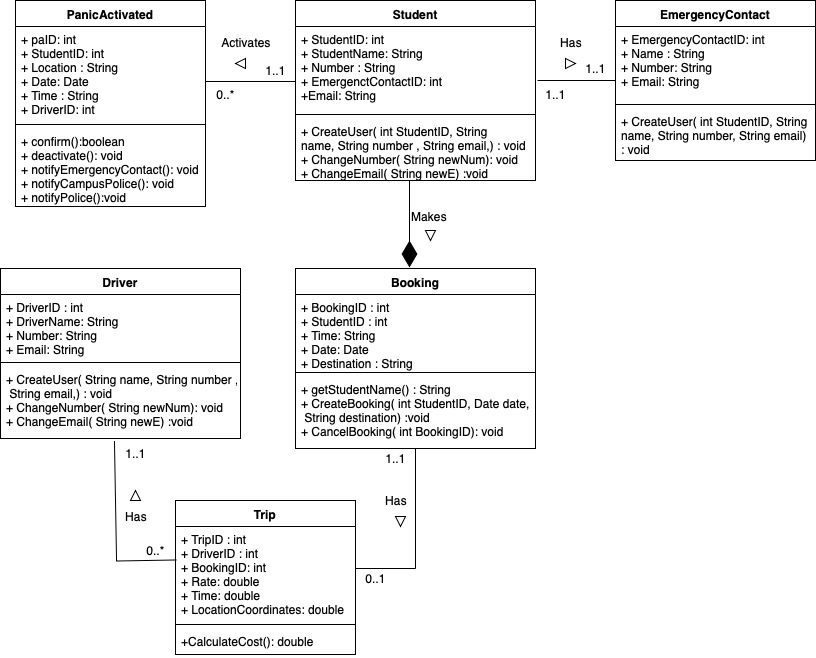
* Models: Student, Driver, Trip, EmergencyContact, PanicActivated
* View: ActivatePanicButtonView (PanicButton widget, message prompt, live GPS, estimated time of arrival, deactivate button, confirm button)
* ActivatePanicButtonPresenter (int paID, int studentID, string location, Date date, String time, int driverID, confirm(),deactivate(), notifyEmergencyContact(), notifyCampusPolice(), notifyPolice())

**USE CASE DIAGRAM**



**Figure 1: Use Case Diagram for UWIBUS application.**

**CLASS DIAGRAM FOR OVERALL SYSTEM**



**Figure 2: Class Diagram for overall system of UWIBUS Application**

**SEQUENCE DIAGRAM**

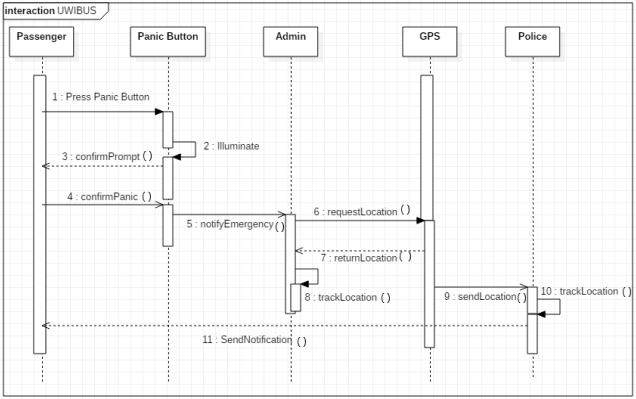


Figure 3: Sequence Diagram for Activating Panic Button.

**PROOF OF METHODOLOGY**

Scrum is an agile method that focuses on managing iterative development.

***Adopting SCRUM:***

* Ideally, a scrum is a daily, short face-to-face meeting of the Scrum team that reviews progress and prioritizes work to be done that day which involves the whole team. However, due to the group’s members various class schedules, we are unable to meet face-to-face daily to conduct this meeting. Therefore, we decided to meet face-to-face on Wednesdays 11am-12noon weekly and discuss via WhatsApp daily, the progress and product backlog.
* The ScrumMaster is responsible for ensuring that the scrum process is followed and guides the team in the effective use of Scrum. We decided that we can rotate ScrumMaster for each sprint. A sprint is a development iteration which is usually 2-4 weeks long.

***Testing Methodologies:***

We plan to use Functional Testing which is split into four components:  unit testing, integration testing, system testing and acceptance testing.

* Unit Testing is defined as a type of software testing where individual units/ components of a software are tested. An automated approach will be used where the team will create test cases for the functions so that whenever a change causes an error, it can be quickly identified and fixed. An advantage to unit testing, is that we can test parts of the project without waiting for other modules to be completed.
* Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group. Our group will apply the Bottom-Up strategy while testing our app. Bottom up integration testing uses test drivers to drive and pass appropriate data to the lower level modules and as and when the code for the other module gets ready, these drivers are replaced with the actual module. An advantage is that the lower level units provide utility to the rest of the application which contest the foundations and high deployment coverage of the software in early phases however early release testing is still limited.
* System Testing, the entire system is tested as per the requirements. It is a Black-box type Testing that is based on overall requirement specifications and covers all the combined parts of a system. Our application is heavily UX based so we will mainly focus on Usability Testing which is catered to the user's ease to use the application, flexibility in handling controls and ability of the system to meet its objectives.
* Acceptance Testing is the process of verifying whether the end to end the flow of the system is as per the business requirements or not and if it suits the needs of the end-user. ACA engineers will utilise the Alpha and Beta Testing method. Alpha Testing normally takes place in the development environment and is usually done by internal staff. Based on the feedback collected from the alpha testers, the team will fix certain issues and improve the usability of the product. After this step, the Beta Testing will commence, this takes place in the customer’s environment and involves some extensive testing by a group of customers who use the system in their environment. These beta testers then provide feedback, which in turn leads to improvements of the product. The client will accept the software only when all the features and functionalities work as expected. It is the last phase of the testing.

**SCRUM METHODOLOGY**

The sprint meeting minutes as well as the Product Backlog and Sprint Backlog were accessed by all group members via Google Docs. The sprint discussions/planning also took place on Google Docs at the beginning of each week before the actual meeting to plan and eliminate the wastage of time. This is because only one meeting can be held each week because of time constraints. Therefore, SCRUM meetings were held once per week on Wednesday 11-12 noon to discuss and make notes, drawings of designs for the intended sprint. These meetings took place in person at the Engineering Undercroft. All team members were able to attend except in one or two unfortunate circumstances. The review of the sprint was done at the end of the week which gave enough time to document the information properly when required. These meetings were aimed at accomplishing each sprint and the details are as follows.

**SPRINT 1:**

21st October – 25th October 2019

**Database and Registration:**

* The system shall allow the user/passenger to register for the application.
* The system shall be linked to a database.

**PRODUCT BACKLOG**

|  |  |  |
| --- | --- | --- |
| **User Story** | **Priority** | **Estimation** |
| Set up a database for the system. | 1 | 4 |
| As a passenger, I would like to register for the application. | 2 | 8 |
| As a passenger, I would like to search/enter a destination to be matched with the closest driver. | 3 | 9 |
| As a passenger, I would like to view the total cost of trip together with all the booking details after booking. | 4 | 9 |
| As a passenger, I would like to activate panic button if I feel unsafe. | 5 | 5 |
| As a passenger, I would like to leave a review of the driver. | 6 | 4 |

**SPRINT BACKLOG**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Story** | **Tasks** | **Day 1** | **Day 2** | **Day 3** | **Day 4** | **Day 5** |
| Setup a database for the system | Create a MSQL database | 1 | 0 | 0.5 | 0 | 0 |
|  | Code Student class | 1 | 1 |  | 0 | 0 |
|  | Test locally | 0 | 0 | 1 | 0 | 0 |
| As a passenger, I would like to register for the application. | Design UI | 1 | 1 | 0 | 0 | 0 |
|  | Code form | 0 | 0 | 2 | 0 | 0 |
|  | Link to database | 0 | 0 | 0 | 3 | 1 |

**SPRINT 1 PLANNING**

**Date:** Monday21st October 2019

**Product Owner:** Anissa

**Scrum Master:** Celine

**User Stories/Tasks:**

* As a passenger, I would like to register for the application.
* Setup database for the system.

**How we plan to work together:**

* Use online chats such as WhatsApp
* Use Google Docs to share and edit the Product and Sprint Backlog
* Use GitHub to collaborate on code

**Design of Registration form:**

**SPRINT 1 REVIEW**

**Date:** Friday 25th October 2019

**Status of User Stories/Tasks:**

* As a passenger, I would like to register for the application - COMPLETED
* Setup database - COMPLETED

**Comments from Product Owner:**

This was the first sprint of the project so I tried to ensure that the team clearly understood the product backlog chosen for this sprint. This sprint is essential as personal information and user credentials are required for safety of passengers and in cases of emergencies. Also, this sprint collected the user’s credentials that will be used for other functionalities that the system will provide. In my opinion, the team understood what was required of them for this increment and it was implemented well. Any misunderstandings were cleared up by the scrum master who sought my input for any clearance.

**Comments from Scrum Master:**

Firstly, I organized the scrum meeting on Wednesday 11-12 noon at Engineering Undercroft. I also ensured the development team understood the requirements of SCRUM Methodology. Additionally, I correlated with the product owner when required to ensure the implementation met the intended vision for this sprint.

**Comments from Development Team:**

The implementation process required the use of trial and error as it was the team’s first time using this platform. Research took an unexpectedly long period of time. Connecting to the database was an issue at first, however, the team overcame it. Additionally, it took serious discipline to follow the SCRUM Methodology including updating the sprint backlog. Advice was frequently taken from the scrum master and the team members now feel more comfortable after this first sprint.

**Inspection (Comments from User):**

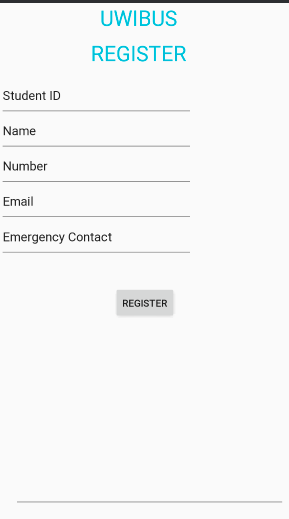
Overall, the user was pleased with the design of the Registration interface as it was easy to understand and the process to enter information was self explanatory.

**Sprint 1 Retrospective:**

The team agreed to do additional research on weekends to ensure it will not hinder development and to get accustomed with the software.

**Implementation of Registration Form:**

*Figure 1.1: Showing registration user interface.*



**SPRINT 2:**

28th October – 1st November 2019

**Booking and Viewing Cost of Trip in Booking Details:**

* The system shall allow passengers to search for rides based on their location and intended destination to be matched with the closest driver.
* The system shall allow passengers to view the total cost of trip together with all the booking details after booking.

**PRODUCT BACKLOG**

|  |  |  |
| --- | --- | --- |
| **User Story** | **Priority** | **Estimation** |
| Set up a database for the system. | 1 | 4 |
| As a passenger, I would like to register for the application. | 2 | 8 |
| As a passenger, I would like to search/enter a destination to be matched with the closest driver. | 3 | 9 |
| As a passenger, I would like to view the total cost of trip together with all the booking details after booking. | 4 | 9 |
| As a passenger, I would like to activate panic button if I feel unsafe. | 5 | 5 |
| As a passenger, I would like to leave a review of the driver. | 6 | 4 |

**SPRINT BACKLOG**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Story** | **Tasks** | **Day 1** | **Day 2** | **Day 3** | **Day 4** | **Day 5** |
| As a passenger, I would like to search/enter a destination to be matched with the closest driver. | Design UI | 1 | 0 | 0.5 | 0 | 0 |
|  | Code the ‘Enter Location and Destination’  selection dropdown button | 0 | 1 | 0.3 | 0 | 0 |
|  | Code a Calendar and time widget | 0 | 0 | 1 | 0 | 0 |
|  | Code a search button. | 0 | 0 | 0.3 | 0 | 0 |
|  | Code the Available Driver selection, when selected | 0 | 0 | 0 | 2 | 0.5 |
|  | Write test cases | 2.5 | 0 | 0 | 0 | 0 |
| As a passenger, I would like to view the total cost and booking details for ride after booking. | Design UI | 1 | 0 | 0 | 0 | 0 |
|  | Create buttons and functions to select desired drivers. | 0 | 1 | 1 | 0.5 | 0 |
|  | Code details to calculate price | 0 | 0 | 2 | 0 | 0 |
|  | Code confirm button | 0 | 0 | 0.5 | 1 | 0 |
|  | Write test cases | 2 | 0 | 0 | 0 | 0 |

**SPRINT 2 PLANNING:**

**Date:** Monday 28th October 2019

**Product Owner:** Anissa

**Scrum Master:** Aakil

**User Stories/Tasks:**

* As a passenger, I would like to search for rides based on my location and intended destination to be matched with the closest driver.
* The system shall allow the user/passenger to view the total cost of the trip as well as all the booking details after selecting the driver.

**Other Goals for Sprint 2:**

* We intend to utilize Extreme Programming (XP) practices such as Continuous Integration, Test-Driven Development (TDD) and Pair Programming.

**SPRINT 2 REVIEW**

**Date:** Friday 1st November 2019

**Status of User Stories/Tasks:**

* As a passenger, I would like to search for rides based on my location and intended destination to be matched with the closest driver - COMPLETED
* The system shall allow the user/passenger to view the total cost of the trip as well as all the booking details after selecting the driver - COMPLETED

**Comments from Product Owner:**

This sprint involved the core functionality of this application which would be non-existent without it. The scrum master and I often conversed as this was an essential feature of the application. I explained that I would prefer the interface to be user friendly so that anyone who wishes to use the application would easily understand what is to be done to make their booking successfully and efficiently.

**Comments from Scrum Master:**

The SCRUM Methodology is being followed closely and diligently and subsequently, progress is being made as it becomes easier to understand and follow. I frequently checked with the development team to ensure that each member was aware of what is required of them for this sprint since everyone was involved in other tasks for their variety of courses. This week we utilized the extreme programming (XP) practices such as continuous integration and pair programming. The team managed to overcome the difficulties of various other tasks and were able to complete the sprint time efficiently.

**Comments from Development Team:**

The advice and check up from the scrum master were drastically important at this hectic week in accomplishing the goals of this sprint. The use of pair programming proved successful as it was easier to discover errors and refactor their pair’s code which also assisted in completing this sprint in a timely manner. TDD was utilized in the sense where difficult test cases were written so the test cases would fail. The team then refactored the code so that all test cases were successfully validated. During this process, Continuous Integration was fundamental as it made the deployment and testing of code easier and efficient.

**Inspection (Comments from User):**

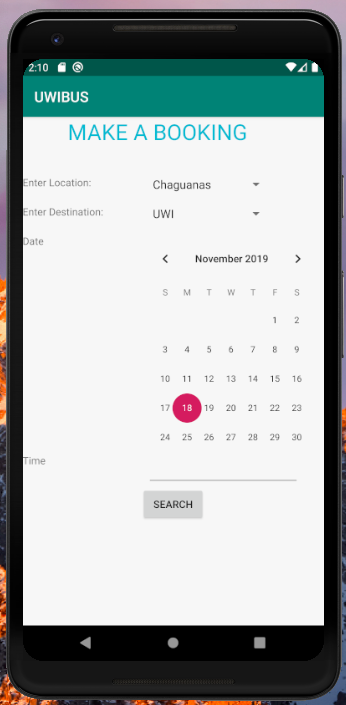
The user found the application simple to use, they appreciated the calendar widget as compared to typing in the date. They also liked that the driver’s reviews were next to the driver’s profile so they can choose the best driver that is available at that point in time.

**Sprint 2 Retrospective:**

The team will manage their time better and they intend to improve the test cases to cover more functionalities.

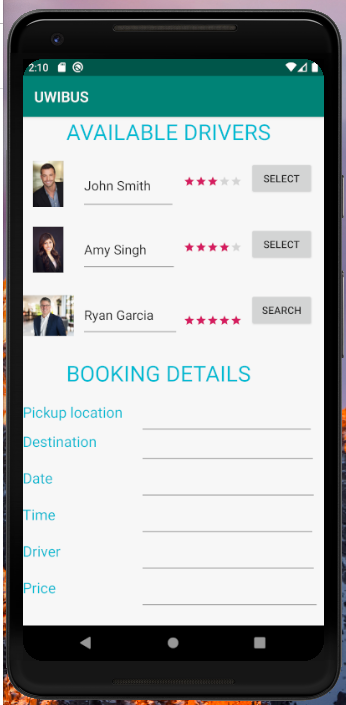
**Implementation of Make a Booking:**

*Figure 2.1: Showing the make a booking user interface.*

****

**Implementation of Available Driver and Booking Details:**

*Figure 2.2 Showing available driver user interface and the booking details which includes the cost of the trip.*



**SPRINT 3**

4th November – 8th November 2019

**Panic button and Review of Driver:**

* The system shall allow the passenger to leave a review of the driver.
* The system shall allow passengers to activate the panic button if he/she feels threatened throughout the trip.

**PRODUCT BACKLOG**

|  |  |  |
| --- | --- | --- |
| **User Story** | **Priority** | **Estimation** |
| Set up a database for the system. | 1 | 4 |
| As a passenger, I would like to register for the application. | 2 | 8 |
| As a passenger, I would like to search/enter a destination to be matched with the closest driver. | 3 | 9 |
| As a passenger, I would like to view the total cost of trip together with all the booking details after booking. | 4 | 9 |
| As a passenger, I would like to activate panic button if I feel unsafe. | 5 | 5 |
| As a passenger, I would like to leave a review of the driver. | 6 | 4 |

**SPRINT BACKLOG**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Story** | **Tasks** | **Day 1** | **Day 2** | **Day 3** | **Day 4** | **Day 5** |
| As a passenger, I would like to activate the panic button if I feel unsafe. | Design UI | 1 | 0 | 0 | 0.2 | 0 |
|  | Code panic button | 0 | 1 | 0 | 0 | 0 |
|  | Code display message | 0 | 0 | 0.3 | 0 | 0 |
|  | Link model, view, presenter | 0 | 0 | 2.5 | 0 | 0 |
| As a passenger, I would like to leave a review of my driver | Design UI | 1 | 0 | 0 | 0 | 0 |
|  | Create a form for review | 0 | 1.5 | 1 | 0 | 0 |
|  | Create a ‘Submit Form’ button | 0 | 0.2 | 0 | 0 | 0 |

**SPRINT 3 PLANNING**

**Date:** Monday 4th November 2019

**Product Owner:** Anissa

**Scrum Master:** Celine

**User Stories/Tasks:**

* As a passenger, I would like to activate panic button if I feel unsafe throughout the trip.
* As a passenger, I would like to leave a review of the driver.

**Other Goals for Sprint 3:**

* Utilize Test Driven Development (TDD)
* Extreme Practices (XP) such as Continuous Integration (CI)

**SPRINT 3 REVIEW**

**Date:** Friday 8th November 2019

**Status of User Stories/Tasks:**

* As a passenger, I would like to activate panic button if I feel unsafe throughout the trip - COMPLETED
* As a passenger, I would like to leave a review of the driver - COMPLETED

**Comments from Product Owner:**

The increment selected for this sprint was intended to complete an important feature which is a panic button. When this button is activated, data such as location and contact information is sent to admin and the TTPS who immediately is dispatched to the passenger’s location. This component ensures the safety of passengers and would therefore be a necessary component in our implementation. While completing this sprint, I emphasized to the development team and scrum master about the necessity of adhering to the principles of software engineering.

**Comments from Scrum Master:**

The team is following good software engineering principles thus far. This is the third week of adhering to these principles and it is becoming easier to adapt daily. The main hindrance is the hectic schedules of the team as they are engrossed in multiple midterm exams and assignments, but they are overcoming this and functioning appropriately.

**Comments from Development Team:**

Thus far, the application has all the core functionality on the passenger’s view of the application. The team has adapted well to the new environment and works continuously to connect all the components of the architecture of the Model-View-Presenter. We often discuss problems encountered with our pairs online. Less documentation is required as we continue along as increments are added continuously. The team has a deep understanding of continuous integration of all the functionalities.

**Inspection (Comments from User):**

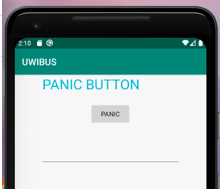
An unnamed user found this component extremely comforting for this implementation. She made recommendations that it would be useful to cancel the activation of the panic button in cases where it was accidentally pressed and if the passenger is not in any threat.

**Sprint 3 Retrospective:**

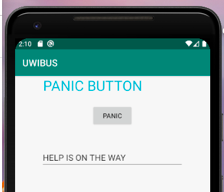
The team aims to make the UX simpler and efficient so that the average user will have an easier time navigating the app.

**Driver Review Implementation:**

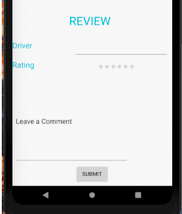
*Figure 3.2 Showing the panic button user interface.*

****

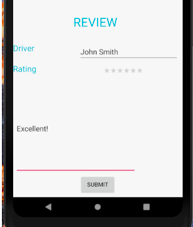
*Figure 3.3 Showing the panic button user interface (when the panic button is clicked).*

****

*Figure 3.4 Showing the driver review user interface.*

****

*Figure 3.5 Showing the driver review user interface (leaving a review)*

****

**SPRINT 4**

11th November – 15th November 2019

**Refactoring and Contingency:**

* All

**PRODUCT BACKLOG**

|  |  |  |
| --- | --- | --- |
| **User Story** | **Priority** | **Estimation** |
| Set up a database for the system. | 1 | 4 |
| As a passenger, I would like to register for the application. | 2 | 8 |
| As a passenger, I would like to search/enter a destination to be matched with the closest driver. | 3 | 9 |
| As a passenger, I would like to view the total cost of trip together with all the booking details after booking. | 4 | 9 |
| As a passenger, I would like to activate panic button if I feel unsafe. | 5 | 5 |
| As a passenger, I would like to leave a review of the driver. | 6 | 4 |

**SPRINT BACKLOG**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User Story | Tasks | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 |
| All | Refactoring code | 2 | 1 | 0 | 1 | 0 |
|  | Add images/backgrounds | 0 | 0 | 0 | 1 | 0 |
|  | Additional enhancements | 0 | 0 | 1 | 1 | 0 |

**SPRINT 4 PLANNING**

**Date:** Monday 11th November 2019

**Product Owner:** Anissa

**Scrum Master:** Aakil

**User Stories/Tasks:**

* All - COMPLETED

**Other Goals for Sprint 4:**

* Extreme Programming Practices (XP) such as Code Refactoring.
* This sprint aims to refactor code and implement aesthetic design to enhance the overall view of the application.

**SPRINT 4 REVIEW**

**Date:** Friday 15th November, 2019

**Status of User Stories/Tasks:**

* All – COMPLETED

**Comments from Product Owner:**

Currently, the application has all the prioritized functionalities. The application is moving along coherently and the increments have progressed successfully.I ensured that the team understood that the interface should be as user-friendly as possible. Therefore, the main goal of this sprint was to enhance the front-end of the software so that anyone who uses the interface can easily understand how to operate it.. The back-end database was also improved in order to make it easier to create or add further iterations.

**Comments from Scrum Master:**

The Scrum Methodology greatly assisted in the completion of all the increments time efficiently. In this sprint, we refactored some code that was loose and since this sprint was utilized for the overall enhancements, all the user stories were involved. However, this was done in a logical sequence.

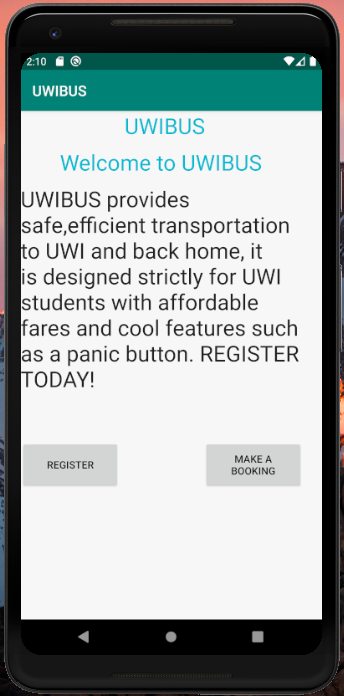
**Comments from Development Team:**

At this stage of the project, it is easier to traverse through the code as the team is comfortable with the environment. The front-end and back-end were well developed for the overall look and the ease of adding iterations respectively.

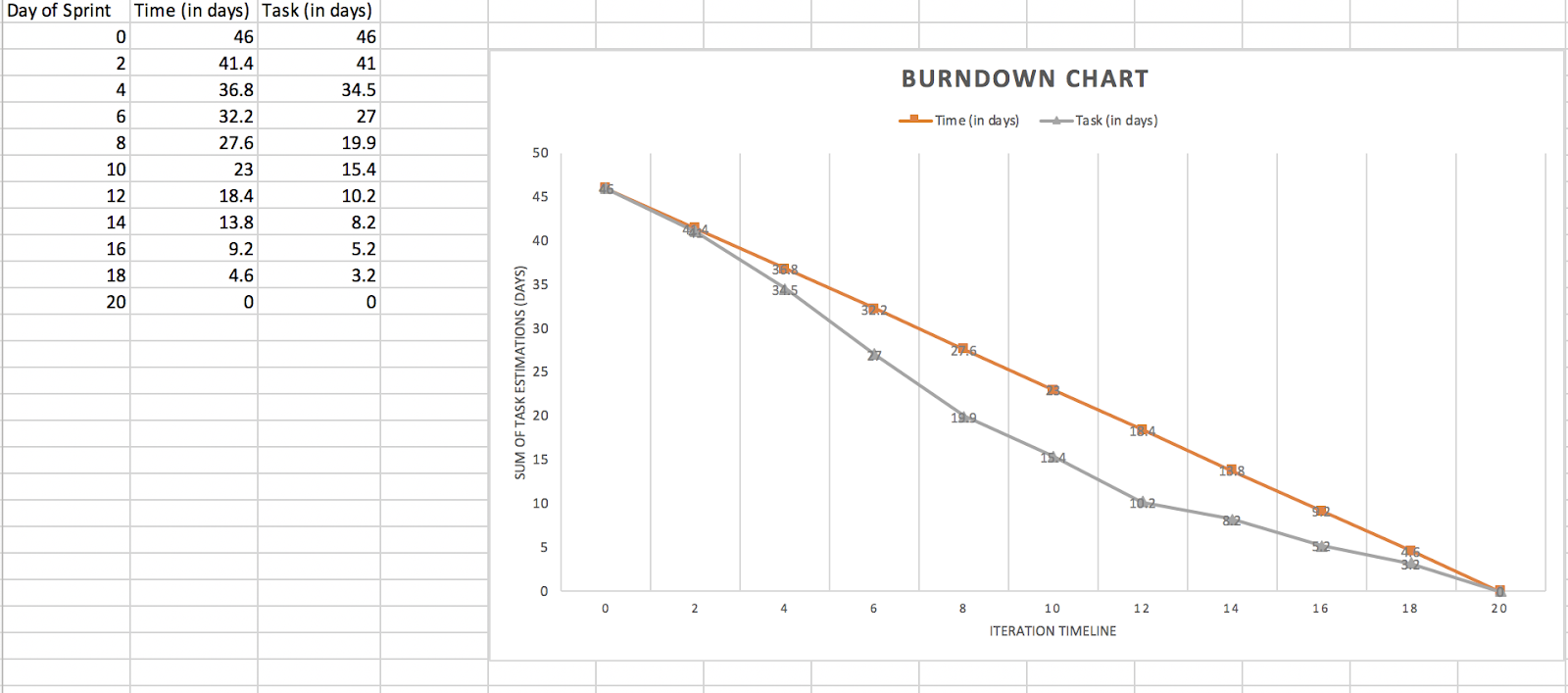
**Inspection (Comments from User):**

The users thought that the end-product reflected exactly what was required from the user stories. They related that the software was properly developed and included all the functionality which would make this application valuable and safe for use whenever necessary. The user is satisfied with the completed enhancement of the application and believed that it was user-friendly. Overall, the user is greatly pleased with the end-product.

*Figure 4.0: Showing the homepage.*



**BURNDOWN CHART**

****

As seen in the chart above, the team worked well together to get all the deliverables done on time. The amount of effort put in enabled the project to be done ahead of schedule at times.

**GITHUB REPO**

<https://github.com/aakilr/SWE-Project-UWIBUS.git>

**GLOSSARY**

**Continuous Integration**- refers to a software development practice requiring developers to integrate code into a central repository at various times of the day.

**GitHub-** an American company that provides hosting for software development. Github includes version control using Git, which allows software projects to keep track of all versions and revert to previous versions if necessary.

**Pair programming-** a development situation where programmers work in pairs, rather than individually to develop code. A fundamental part of extreme programming.

**Refactoring-** modifying a program to improve its structure and readability without changing its functionality.

**Risk management**- the process of identifying risks, assessing their severity, planning measures to put in place if the risks arise and monitoring the software and the software process for risks.

**SCRUM**- an agile method of development which is based on sprints- short development cycles. Scrums may be used as a basis for agile project management alongside other agile methods such as XP.

**Test Driven Development (TDD)-** an approach to software development where executable tests are written before the code.

**User story-** a natural language description of a situation that explains how a system/s might be used and the interactions with the system/s that might take place.

**Validation**- the process of checking that a system meets the needs and expectations of the customer.

**Version control-** a system that records changes to a file or set of files over time so that you can recall specific versions later.

**REFERENCES**

Sommerville, Ian. “Software Engineering.” Pearson, 2015.

Mike Cohn. User Stories. [www.mountingoatsoftware.com/agile/user-stories](http://www.mountingoatsoftware.com/agile/user-stories).

Visual Paradigm. Sequence Diagram. <https://www.visual-paradigm.com/guide/uml-unifiedmodeling-language/what-is-sequence-diagram/>

Visual Paradigm. Class Diagram. <https://www.visual-paradigm.com/guide/uml-unified-modelinglanguage/what-is-class-diagram/>

Visual Paradigm. Continuous Integration. <https://www.visual-paradigm.com/scrum/scrum-continuous-integration-delivery-deployment/>

Github. <https://github.com/>