**IS4301 Agile IT with DevOps 2022**

**Open Book Assignment**

|  |  |
| --- | --- |
| **Name:** | **Le Van Minh** |
| **Matric number:** | **A0219739N** |
| **Email:** | **vanminh09@u.nus.edu** |
| **Sprint team:** | **ScrumdogTrillionaire** |

***Instructions:*** *Complete the questions below using this document and name your file “<<Your name>> IS4301 2022 Assignment.doc”. Upload your completed assignment into LumiNUS Assignment Submission folder with time stamp no later than 0800 hours Nov 4th 2022. Only one submission is to be uploaded. There is a penalty of 20% of total mark for < 2 days of late submission, 40% penalty for 3-7 days. No submission will be accepted after 11th November 2022.*

*Gentle reminder regarding NUS Code of Conduct and Plagiarism:* [*http://www.cdtl.nus.edu.sg/ta-handbook/academic-integrity.htm*](about:blank)*. It is important to note that****plagiarism****is an academic offence that is taken very seriously by the University, as stated in the****NUS****Code of Student Conduct.*

This open book assignment constitutes 40% of your total grade for this course.

Composition of Questions:

|  |  |  |
| --- | --- | --- |
| **Questions** | **Allocated Marks** | **Actual Marks** |
| 1 (Iterative planning) | 10 |  |
| 2 (Estimations) | 14 |  |
| 3 (User story development) | 20 |  |
| 4 (Microservices Architecture) | 22 |  |
| 5 (Coding – gitflow) | 5 |  |
| 6 (Coding – client side) | 15 |  |
| 7 (Coding – server side) | 10 |  |
| 8 (Branching) | 4 |  |
| TOTAL | 100 |  |

**Question 1 (10 marks)**

1. Suppose an Epic consists of estimates for eight user stories as shown in below table.
   1. Complete the column Bang-for-the-Buck. (2 marks)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Story Points**  **(estimated by developers)** | **Value Points**  **(estimated by product owner)** | **Bang-for-the-Buck** |
| **User Story 1** | 34 | 144 | 4.2353 |
| **User Story 2** | 21 | 55 | 2.6190 |
| **User Story 3** | 55 | 89 | 1.6182 |
| **User Story 4** | 13 | 21 | 1.6154 |
| **User Story 5** | 34 | 55 | 1.6176 |
| **User Story 6** | 2 | 2 | 1.0000 |
| **User Story 7** | 8 | 2 | 0.2500 |
| **User Story 8** | 55 | 1 | 0.0182 |

* 1. The sprint team who implements this epic has a velocity of 55 story points over a two-week sprint. Suppose there is an implementation dependency where user story 3 must commence first over the rest of user stories, complete the table below for planned iterations in implementing this epic. Add a row to the table below for each planned iteration to complete this epic. (4 marks)

|  |  |  |  |
| --- | --- | --- | --- |
| **Planned Iteration** | **User Stories in Iteration** | **User Story Points for this Iteration** | **Value Points for this Iteration** |
| 1 | User Story 3 | 55 | 89 |
| 2 | User Story 1, 2 | 55 | 199 |
| 3 | User Story 4, 5, 6 | 49 | 78 |
| 4 | User Story 7 | 8 | 2 |
| 5 | User Story 8 | 55 | 1 |

* 1. Must all iterations be implemented to complete implementation of this epic? Explain your reason with respects to your planned iterations above. (2 marks)

|  |  |
| --- | --- |
| **Answer:** | Only the first three iterations are required to complete this epic. At the end of the 3rd iteration, 366 value points is fulfilled, over the total 369 points for the whole epic (> 99%). So the epic is practically completed. Furthermore, the remaining 1% value points account for 63 story points (> 28%), which is not worthwhile. |

1. Draw the burn-down chart for this epic. (2 marks)

|  |  |
| --- | --- |
| **Burn-Down Chart:** |  |

**Question 2 (14 marks)**

1. In an epic, Iteration 7, which consists of the following two user stories, was completed over 10 days in a two-week sprint:

|  |  |  |
| --- | --- | --- |
|  | **Story Points** | **Value Points** |
| **User Story 71** | 5 | 13 |
| **User Story 72** | 8 | 8 |

* 1. What is the velocity of Iteration 7? Show your calculation. (2 marks)

|  |  |
| --- | --- |
| **Answer:** | Velocity: 13  ---  Assume a 5-day-working week.  Time consumed: t = 1 (week)  Total story points completed: p = 5 + 8 = 13 (story points)  Velocity: v = p/t = 13/1 = 13 (story points/sprint) |

* 1. How long is a story point in hours for this Iteration? Show your calculations. (2 marks)

|  |  |
| --- | --- |
| **Answer:** | 1 story point = 6.15 hour  ---  Assume an 8-hour-working day.  Total hours consumed: h = 10 \* 8 = 80 (hours)  Total story points completed: p = 5 + 8 = 13 (story points)  h/p = 80/13 = 6.15 (hours/point) |

After completion of Iteration 7, a backlog of 6 user stories remains:

|  |  |  |
| --- | --- | --- |
|  | **Story Points** | **Value Points** |
| **User Story 81** | 5 | 5 |
| **User Story 82** | 8 | 3 |
| **User Story 83** | 3 | 2 |
| **User Story 84** | 8 | 3 |
| **User Story 85** | 5 | 2 |
| **User Story 86** | 21 | 3 |

* 1. Which user stories will be planed for in Iteration 8? Explain your answer. (4 marks)

|  |  |
| --- | --- |
| **Answer:** | User stories planned for Iteration 8: User Story 81, 83, 85  ---  Bang-for-the-buck score:  User Story 81: 1.00  User Story 82: 0.38  User Story 83: 0.67  User Story 84: 0.38  User Story 85: 0.40  User Story 86: 0.14  Ranked for highest-to-lowest BFTB: 81, 83, 85, 82, 84, 86.  Our best estimate for velocity, based on Iteration 7 is 13 (story point/sprint)  The 3 highest ranked user stories sum up to 13 story points, quick is just equal to estimated velocity.  Therefore, User Story 81, 83, 85 will be planned for Iteration 8. |

* 1. How long is the backlog of these 6 user stories in terms of remaining user story points, number of hours required to implement this backlog, and estimated number of iterations needed after Iteration 7? Show your calculations. (6 marks)

|  |  |
| --- | --- |
| **Answer:** | Story Points remaining: 50 (story points)  Total story points = 5+8+3+8+5+21 = 50 (story point)  Estimated number of hours required for this backlog: 307.5 hours  Average hours for each story point: 6.15  Estimated hours for this backlog = 6.15 \* 50 = 307.5  Estimated number of iterations needed to complete backlog: 4  Velocity = 13  Total story points = 50  Estimated number of iterations = 50/13 = 3.85 ≈ 4 |

**Question 3 (20 marks)**

1. From your chosen Fintech Statement in your agile team project,
   1. Depict 5 user stories which you developed personally (10 marks)

|  |  |
| --- | --- |
| Sprint Team: ScrumdogTrillionaire  Fintech Statement: Statement 1 – Banking application encouraging ESG sustainability | |
| User Story 1 | "As a purchaser, I want to pay for my items with QR code scanning, so I can pay conveniently any time with my phone."  ---  Story point: 8  Value point: 13  ---  Acceptance criteria:  - When a store staff/checkout machine generate a transaction, then a QR code attached to the transaction is displayed.  - Given a QR for a transaction is generated, and purchaser has logged into the app, when the purchaser scans the QR code and confirm payment, money should be transferred from the purchaser to the seller. |
| User Story 2 | "As a purchaser, I want to view all my past transactions, so I can better track my spending."  ---  Story point: 2  Value point: 2  ---  Acceptance criteria:  - Given when a user has logged into the app, when the user opens the transaction history, then a list of past transaction should be displayed. |
| User Story 3 | "As a purchaser, I want to view the items I have purchased in a particular transaction, so I can know what I have bought."  ---  Story point: 13  Value point: 8  ---  Acceptance criteria:  - Given when a user has logged into the app, when the user open a past transaction, then the details of the transaction (date time, total amount, etc.) should be displayed with list of purchased items and their prices. |
| User Story 4 | "As a purchaser, I want to view the amount of impact I made on the environment through my purchases, so I can assess my consuming habit more critically."  ---  Story point: 13  Value point: 8  ---  Acceptance criteria:  - Given when a user has logged into the app, when the user views their sum ecological impact, then a figure should be displayed indicating the impact of their purchases.  - Given when a user has logged into the app, when the user view ecological impact of a certain transaction, then a figure indicating the impact of that transaction should be displayed, with impact of individual items if available. |
| User Story 5 | "As a purchaser, I want to view my past purchases grouped by categories, so I can know which activity is eating up my budget."  ---  Story point: 13  Value point: 8  ---  Acceptance criteria:  - Given when a user has logged into the app, when the user opens the consumption summary, a graph/figures indicating the spending on each category (transport, recreation, food, etc.) should be displayed.  - Given when a user has logged into the app, when the user opens the consumption details, then a list of transactions of last month should be displayed and grouped to their respective category. |

* 1. For each user story, explain how each of them satisfies the INVEST principle. (10 marks)

|  |  |
| --- | --- |
| User Story 1 | - Independent: The user story is as atomic as it can be, with specific function and expectation. There should be no dependence on other stories.  - Negotiable: Should the QR be statically or dynamically generated? Can we use other encoding? How about Bluetooth payment? These are questions that can be asked by the team.  - Valuable: The feature allows purchaser to make payment quickly, without having to deploy their cards or cash. The implementation on the seller side is simpler, too, with only a screen to display QR is sufficient.  - Estimatable: The implementation process is well understood, and the technology is not alien, so the team should be able to estimate it well.  - Small: QR code scanner, payment API, QR code generator are possible technical feats. This should fit well in a two-week sprint.  - Testable: Very clear acceptance criteria can be formed. |
| User Story 2 | - Independent: It does not depend on any other features, as it is a simple data obtaining process.  - Negotiable: The team can discuss on how far back they should be able to track their transactions.  - Valuable: The user can check if they have made a certain payment, e.g., this week groceries  - Estimatable: A very basic and simple use feature, should be well understood among the team  - Small: A very small, frontend feature  - Testable: can form criteria easily |
| User Story 3 | - Independent: Quite independent from other features, as viewing a particular transaction can be done without being able to view all transactions, for example  - Negotiable: The support can be provided for the feature is debatable. For example, to what extend are we dependent on the compliance from our service user (i.e., the retailer using our payment system) to successfully implement this function?  - Valuable: Not only the consumer can monitor their own shopping habit, but we can also develop recommendation system from these data.  - Estimatable: Although the feature is quite uncommon among banking app, there's no novel technology involve. Estimating the workload should be trivial.  - Small: The story point is quite big comparing to others, but it should be doable in one sprint.  - Testable: The accepting criteria can be straightforwardly determined. |
| User Story 4 | - Independent: This feature does not depend on any other features.  - Negotiable: The team can debate on how the "impact" be presented; how would they calculate the impact.  - Valuable: The user can make use of this feature to be more conscious about the environment, which is a greater good for everyone.  - Estimatable: Although the feature seems uncommon, the technology should not be new, and the framework to assess the impact should have been well studied by ecologist.  - Small: The estimated story point indicate that it should fit well into one sprint.  - Testable: Testing criteria can be formed systematically. |
| User Story 5 | - Independent: This feature should not depend on any other feature.  - Negotiable: The team can discuss which categories are included, which graph/figures should be displayed.  - Valuable: consumption composition is a very useful information for personal budgeting.  - Estimatable: The feature is nothing technically challenging, we should be able to estimate it easily.  - Small: The feature is small enough.  - Testable: Constructing testing criteria should be trivial. |

**Question 4 (22 marks)**

1. From the “Online Product Purchase” example on slide 11 of Lecture 6:

|  |
| --- |
| **Online Product Purchase**   * Search online product catalogue * Add product to cart * Check out * Create new account * Specify mode of payment * Specify payment details * Specify email address * Specify mailing address * Complete order * Verify purchase details * Complete card transaction * Print completed order |

1. Draw a possible Service Oriented Architecture design for this Online Product Purchase example. (4 marks)

|  |
| --- |
|  |

1. Draw a possible Microservices Architecture for Online Product Purchase Example: (4 marks)

|  |
| --- |
|  |

1. Compare both of your designs from perspective of an agile implementation. (4 marks)

|  |  |
| --- | --- |
| Service Oriented Architecture | Microservice Architecture |
| * Loose coupling is enabled via independent services * Scaling is not flexible * Each service component is not independently deployable * Not focusing on DevOps and CI/CD * Uses a common platform for deploying all component services | * Loose coupling is enabled via self-contained applications. * Flexible and independent Scaling * Each service component is a self-contained application and is independently deployable * Strong focus on DevOps and CI/CD * Utilizes cloud technologies for more efficient deployment |

1. An additional service called “Select Self-Pickup Delivery Option” is added to the list of current Online Product Purchase services. Develop user story/stories for this new service called “Select Self-Pickup Delivery Option”. (4 marks)

|  |  |
| --- | --- |
| **User Story/User Stories:** | - As a purchaser, I want to see all the available delivery option, so I can choose the best fit to my preference.  - As a purchaser, I want to choose self-pickup as delivery option, so I can collect my items at my leisure.  - As a purchaser, I want to specify self-pickup details, so my collection can be as convenient to me as possible.  - As a purchaser, I want to be emailed a copy of self-pickup details, so I can verify that my request is correctly registered, and I can assess the information later. |

1. Show changes to your microservices architecture resulting from adding this new service. (2 marks)

|  |
| --- |
|  |

1. In order that this new service can be deployed at the end of a two-week sprint, recommend tool chain for continuous development and continuous integration for Online Product Purchase services example. Explain your choice of selection. (4 marks)

|  |  |
| --- | --- |
| **Recommended CI/CD DevOps toolchain with explanations and justifications:** | The CI/CD pipeline is comprised of several stages, where each tool will be employed to accomplish the task:  - **Collaboration:** Jira (application lifecycle management), Slack (Communication).  + Jira is a powerful and flexible management tool with friendly interface.  + Slack has a sophisticated channel management system is ideal for Agile teams.  - **Build:** Git and GitHub (VCS), CircleCI (CI tool), Docker (Build), DBMaestro (Database management).  + Git is a powerful tool with many features and very stable.  + GitHub is a familiar platform to most developers.  + Circle CI is a fully automated, very quick and robust for testing and deployment.  + Docker is very assessable and standardized environments using local containers, allowing building, sharing, and running containerized applications easy for developers of all skill levels.  + DBMaestro is a sophisticated tool, using continuous feedback loop to accelerate release cycles.  - **Test:** MochaJS (Test framework for JavaScript)  + MochaJS is a very popular tool with high compatibility, accurate reporting and fit for both frontend and backend testing.  - **Deploy:** Octopus Deploy (Deployment)  + Octopus Deploy is fully automated and offer very easy management with powerful continuous delivery tools.  - **Run:** Dokku (PaaS)  + Dokku is a PaaS tool that works very well with Docker. Dokku is very small, yet powerful with lifecycle and scaling management. |

**Question 5 (5 marks)**

1. Use Git as a version control tool for the question 6 and 7. Start a develop branch after you create a new react project. Record the changes made for the exercise 6 and 7. Use Git to output the differences between the initial snapshot (first commit) and the latest snapshot when you finish exercise 7. Insert the screen shots of your steps to complete this question.

*Hint: Use command “git diff”*

|  |  |
| --- | --- |
| **Screen Printed Steps and Results:** | - create new React project "is4301-assgn":    - Create ‘develop’ branch, checkout and see the branch log:      - After finishing question 6&7, print the log:  git log --graph --all --decorate --oneline    - Output the difference between the first and last commit (mostly node module imports):  git diff b85dac2 f49d800 |

**Question 6 (15 marks)**

1. Refer to Lab Exercise 1 and create a home page that displays a NUS logo, a text that says: “NUS is a leading research university in Asia”, and a working navigation panel with Home, Academics, Admission. The academics page should have a list of modules taught at the university and the admission page should have an admission process flow.

The list of modules taught at the university should be formatted as a table:

| Module Code | Module Title |
| --- | --- |
| BT1101 | Introduction to Business Analytics |
| BT2101 | Econometrics Modelling for Business Analytics |
| BT2102 | Data Management and Visualisation |
| BT2103 | Optimization Methods in Business Analytics |

The admission timeline should be formatted as a bullet point list:

1. Understand application online
2. Upload supporting documents
3. Make application fee payment
4. Check application status

Screenshot the steps you made to complete the task.

|  |  |
| --- | --- |
| **Screen Printed Steps and Results:** | - Change logo and home screen text (imported NUS logo to src/ directory):    - Change background colour and stop the spinning:      - Refactor to contain ‘Home’ in a separate component:    - Academics component + styling:      - Admission component + styling:      - Import React Router, and config Router in the main App component:      - Import React Bootstrap:    - Create Navigation component (navbar) and styling (with support from Bootstrap):      - Add Navigation to the App component:    - Results: |

**Question 7 (10 marks)**

1. Extend the previous exercise by adding a sign in with Google button to all the pages. Account for the extreme cases, like wrong credentials and non-existent user by triggering a notification message that gives an explanation. Demonstrate that the authentication flow works by signing in with your google account, your account with a wrong password, and an account that does not exists in the database.

*Hint:* *you may reuse some code and firebase settings from the tutorial 2*

|  |  |
| --- | --- |
| **Screen Printed Steps and Results:** | - Create a new Firebase project and register a new Web app:    - Enable Authentication using Google account:    - Enable Firestore, and set assess permission to anyone:      - Install firebase and react-firebase-hooks:    - Use the template provided in Tutorial 7 to config connection with Firebase, enable Google login:    - Add Login with Google button to Navigation bar:      - Conditionally display login button and greeting based on the login status:      - Login with Google account:      - Account with wrong passwords:    - Non-existent account:    - Add Login button to direct to login page:      - Create login page from template provided in tutorial 7, with some styling:    - Create an account with email and password via Firebase console:    - Login with correct credentials:    (Successful login, but the user is not named yet, therefore name=null    - Login with wrong credentials:    - Login with non-existent account: |

**Question 8 (4 marks)**

1. Your e-commerce product team has a regional presence in South-East Asia countries, with usual goods delivery cycle of 5-8 working days. The Product Owner plans to add a new service of self-pickup delivery in one of the countries. One of your squad members estimates that this new feature of self-pickup delivery in this country will require two sprint cycles to complete.
   1. Is trunk-based development more suitable than feature-based development for implementing this new feature? Explain your answer. Further, explain how continuous integration is related to this development. (2 marks)

|  |
| --- |
| Trunk-based development is more suitable than feature-branching. Because two sprints are a long time, during which a big change can happen the code base, or another team is developing a separate feature. If employing feature-branching, the conflict-resolving process will be laborious and inefficient. Trunk-based development ensure the code line is always up-to-date and executable, reducing risk of conflict and distribute merge-error resolves evenly through time and between teams. |

* 1. At the same time with this new feature implementation, a hotfix needs to be developed and applied. Your squad targets to apply this hot fix and the new feature implementation into the next Release cycle in 2 months’ time. Complete the below picture to show these two code developments into Mainline trunk. You may draw the picture separately and paste into the box below. (2 marks)

|  |
| --- |
|  |

END of Pages for Assignment