|  |  |
| --- | --- |
| Team 1 – Lifecycle Justification  and Process Plan | Ivan, Natalia, Ronan  Software Design with AI for Cloud Computing 3.1 |

# Justification

Our team decided to go for Incremental Delivery, reasons why:

1. **Requirements are well defined** - this allows us to divide the project into smaller phases which we will deliver in increments. Our plan was to pick key increments that would be presented to customer so we can gain valuable feedback and ensure needs are met. After further consideration we decided to involve customer all the way through and present each increment to receive valuable feedback and/or ensure needs are met.
2. **Feedback** - as we want to include customer input/feedback after each increment (which will be highlighted in the plan below), incremental delivery will allow us to present work done to customer. This will also ensure we can incorporate changes at earlier stage if needed.
3. **Supports changes to some extent** – Incremental Delivery is a Plan Driven approach that assumes nothing should change. Although, it doesn’t mean the scope cannot change between increments when required. In the lifecycle we choose, the increments are planned in advance although we still have benefit of making small tweaks if necessary.
4. **Development Process** - it is more visible how much has been done and what’s left to do as each increment is completed and delivered. As we will choose key phases that customer will be involved in, it will be reassuring for them.
5. **Phased Development** – development is done sequentially, where each increment is building up on the previous one. This also doesn’t mean we must cross out parallel development. We can have two developers working on two different increments simultaneously. This will ensure deadline is met and if we are ahead of time, we will have either more time for testing or supporting another developer.
6. **Product Availability** – Incremental Delivery allows us to have working software quicker, which can provide value to customer sooner. It for sure will not be a fully working product after first delivery, although some functionality can be available to customer easier (eg. signing up new users into the system)
7. **Testing** – testing is involved with each increment, which ensures high quality of software is delivered to customer. We also are not moving to the next increment until everything is fully working.
8. **Self-organizing** – we are a self-organizing team which means we are not waiting for a manager (or Scrum) to get the work assigned. We manage our responsibilities and timelines ourselves.

We have decided not to go for Agile as requirements are well-defined and we don’t need to involv4e customer constantly. Instead, we will present what we have at its key phases. We also rejected Waterfall due to it’s total lack of flexibility. It is highly sequential model, where each phase is dependent on the previous one. We wouldn’t be able to have two devs working on separate phases as everything happens one after another. Here, we also don’t have customer input until the end which can potentially make it very costly to revise/fix if there are issues. We also rejected V-Model as the team is too small to take that approach. It requires extensive testing and parallel working which we don’t have resources for. It would potentially add extra complexity to the project.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Increment | Task | Description | Assignee | Estimation | Actual | Week | Done Yes/No | Comments |
| 1. Initial Planning 2. Specification 3. Design | Team Setup | Prepare working agreement, create/clone team repo | Natalia Ronan Ivan | 4h x each |  | WK 3 |  |  |
| User Stories | Admin, Newsagent, Customer | Natalia | 3h |  | WK 3-4 |  |  |
| Delivery Driver, Publication (book) | Ronan | 2h |  | WK 3-4 |  |  |
| Invoice, Order | Ivan | 2h | 3h | WK 3-4 |  | Underestimated |
| Design System Plan | Design UML for User Stories | Natalia | 2h  x each | 1h | WK 5 |  | Overestimated |
| Ronan | 3h | WK 5 |  |  |
| Ivan | 1h | WK 5 |  | Overestimated |
| System Architecture | Describe system architecture | Natalia | 2h |  | WK 4 |  |  |
| Create flow chart | Ivan | 1h | 2h | WK 4 |  | Lack of depth understanding , challenging |
| Create sequence diagram | Ronan | 3h |  | WK 5 |  |  |

# Process Plan

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Increment | Task | Description | Assignee | Estimation | Actual | Week | Done Yes/No | Comments |
| Increment 1 | Design Test Cases | Design Tests for MySQL queries and JUnit  Natalia: customer/newsagent/admin  Ronan:  delivery doc/publication  Ivan:  invoice/order | Natalia | 3h |  | WK  4-5 |  |  |
| Ronan | 3h |  | WK  4-5 |  | Delayed due to internship interviews |
| Ivan | 3h | In Progress | WK  4-5 |  | Delayed due to sickness and internship interview |
|  |  |  |  |  |  |  |  |  |
| Increment 2 | Create Database | Create customer, driver/ newsagent/admin database | Natalia | 2h | 1h | WK 6 |  |  |
| Create delivery doc, publication database | Ronan | 3h |  | WK 6 |  | Delayed due to internship interviews |
| Create invoice, order database | Ivan | 3h | 1h | WK 6 |  | With user stories done was easy to proceed… |
|  |  |  |  |  |  |  |  |  |
| Increment 3 | Implementation | Create database, customer, user (newsagent/admin/driver) classes | Natalia | 15h  x  each |  | WK  7-11 |  |  |
| Create delivery doc, publication classes | Ivan |  |  | publication will be done by Sunday and delivery doc will take shorter |
| Create invoice, subscription classes | Ronan |  |  |  |
|  |  |  |  |  |  |  |  |  |
| Increment 4 | Instruction Development | Develop descriptive instruction how the app is used | Natalia | 2h |  | WK 12 |  |  |
| Customer Presentation | Presentation of fully working software to customer | Natalia  Ronan  Ivan | 1h |  | WK 13 |  |  |
|  |  |  |  |  |  |  |  |  |

## Descriptive Process Plan:

Increment 1: Initial Planning, Specification and Design

* Gather user requirements
* Develop User Stories based on gathered requirements

Entities:

* + Newsagent (6 user stories)
  + Delivery Driver (4 user stories)
  + Admin (6 user stories)
  + Invoices (4 user stories)
  + Publication (2 user stories)
  + Subscription (3 user stories)
  + Delivery Doc (4 user stories)

Total: 29 User Stories

* System Architecture
  + Description for system architecture

High level components:

1. Newsagent App (Command-Line UI) represents the command line interface through which user interacts with the system. Users input commands to perform various actions, like managing customer details, processing invoices etc.
2. Java Application (Business Logic) contains the logic of the Newsagent application. It interprets user commands, interacts with the MySQL database and performs operations like data retrieval, processing, updating etc.
3. MySQL Database (Data Storage) is where the database tables is stored. The database contains tables for managing customer details, newsagent information, delivery drivers, delivery docs, invoices, publications and subscriptions. Foreign keys establish relationships between the tables for data consistency.

A diagram of a data flow

Description automatically generated

* + Flowchart

*A diagram of software

Description automatically generated*

A screenshot of a computer screen

Description automatically generated

\*\*\*\*\* SEQUENCE CHART \*\*\*\*\*

# Incremental Delivery Flow

A diagram of a delivery process

Description automatically generated

# Activity Bar Chart

Chart shows the flow of tasks, and dependencies. Eg, T5 is dependent on task T1 and T5 cannot be conducted until T1 is finished.

A screenshot of a computer

Description automatically generated

### Increment 1: Design Tests for MySQL queries and JUnit: Task Distribution

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Assignee | Description | Completed |
| Increment 1 : WK4-5 | | | |
| # T1 | Natalia | Design test cases for JUnit and MySQL querries for customer/newsagent/admin |  |
| # T2 | Ronan | Design test cases for JUnit and MySQL querries for delivery doc/publication/driver |  |
| # T3 | Ivan | Design test cases for JUnit and MySQL querries for invoice/order |  |
| # T4 | Natalia | Review Ivan’s test cases |  |
| # T5 | Ronan | Review Natalia’s test cases |  |
| # T6 | Ivan | Review Ronan’s test cases |  |

### Increment 2: Create database: Tasks Distribution

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Assignee | Description | Completed |
| Increment 2 : WK6 | | | |
| # T7 | Natalia | Design tables for customer/newsagent/admin |  |
| # T8 | Ronan | Design tables for delivery doc/ publication/driver |  |
| # T9 | Ivan | Design tables for invoice/order |  |
| # T10 | Natalia | Review Ronan’s tables |  |
| # T11 | Ronan | Review Ivan’s tables |  |
| # T12 | Ivan | Review Natalia’s tables |  |
| # T13 | Natalia | Test mysql queries for created tables |  |
| # T14 | Ronan | Test mysql queries for created tables |  |
| # T15 | Ivan | Test mysql queries for created tables |  |

### Increment 3: Implementation: Tasks Distribution

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Assignee | Description | Completed |
| Increment 3 : WK 7-9 | | | |
| # T16 | Natalia | Create DatabaseManager class |  |
| # T17 | Natalia | Create Admin class |  |
| # T18 | Natalia | Create Customer class |  |
| # T19 | Ronan | Review Admin class |  |
| # T20 | Ivan | Review Database class |  |
| # T21 | Ivan | Review Customer class |  |
| # T22 | Ronan | Create Publication class |  |
| # T23 | Ivan | Review Publication class |  |
| # T24 | Ivan | Create Order class |  |
| # T25 | Ronan | Review Order class |  |
| # T26 | Natalia | Create Driver class |  |
| # T27 | Ronan | Create Delivery Doc class |  |
| # T28 | Natalia | Create Newsagent class |  |
| # T29 | Ivan | Create Invoices class |  |
| # T30 | Ivan | Review Delivery Doc class |  |
| # T31 | Ivan | Review Newsagent class |  |
| # T32 | Natalia | Review Invoices class |  |
| # T33 | Ronan | Review Driver class |  |
| # T34 | Natalia | Link, Integrate & Test   1. Admin + User 2. Newsagent + User 3. Driver + User 4. Newsagent + Customer |  |
| # T35 | Ivan | Link, Integrate & Test   1. Customer + Order 2. Publication + Order 3. Delivery Doc + Order 4. Invoice + Customer 5. Invoice + Publication |  |
| # T36 | Ronan | Link, Integrate & Test   1. Customer + Delivery Doc 2. Invoice + Delivery Doc 3. Publication + Delivery Doc 4. User + Delivery Doc |  |
| # T37 | Natalia/Ronan/Ivan | Test the whole application |  |

**DATABASE DESIGN**

**Testing:**

1. Run various SQL queries to retrieve/insert/manipulate data
2. Static Test done by another developer

**Database Tables:**

* CustomerDetails
  + custID (Primary Key) INT AUTOINCREMENT
  + firstName VARCHAR (10)
  + lastName VARCHAR (10)
  + custAddress VARCHAR (15)
  + phoneNo VARCHAR (13)
* UserDetails (newsagent/driver/admin)
  + userID (Primary Key) INT AUTOINCREMENT
  + username VARCHAR (10)
  + password VARCHAR (255)
  + role ENUM (“admin”, “newsagent”, “driver”)
* DeliveryDriver
  + driverID INT AUTO\_INCREMENT PRIMARY KEY,
  + userFK INT,
  + deliveryDistrict VARCHAR(15)
  + FOREIGN KEY (userFK) REFERENCS userdetails (userID)
* DeliveryDoc
  + docID (Primary Key) INT AUTOINCREMENT
  + custID INT
  + publicationID INT
  + dateDelivered DATE
  + FOREIGN KEY (custID) references Customer\_Details(custID)
  + FOREIGN KEY (publicationID) references Publication(publicationID)
* Invoices
  + invoiceID (Primary Key) INT AUTOINCREMENT
  + custID INT
  + publicationID INT
  + dateDelivered DATE
  + totalSum DECIMAL
  + invoiceDate DATE
  + FOREIGN KEY (custID) references CustomerDetails(custID)
  + FOREIGN KEY (publicationID) references Publication(publicationID)
  + FOREIGN KEY (docID) references DeliveryDoc(docID)
* Publication
  + publicationID (Primary Key) INT AUTOINCREMENT
  + title VARCHAR
  + price DECIMAL
  + quantity INT
  + docID INT
  + FOREIGN KEY (docID) references DeliveryDoc(docID)
* Subscription
  + Relationship -> Customer Details -> Retrieve all details
  + custID
  + subscriptionType VARCHAR
  + custStatus VARCHAR
  + deliveryDisctrict INT
  + startDate DATE
  + endDate DATE
  + FOREIGN KEY (custID) references CustomerDetails(custID)