Antenatal Care Iteration 1 Report

**Team Leader: Nicholas Curlew**

**Team 1 (Frontend): Ross McLain, Naomi Heuer**

This team focused primarily on the User Interface, which included three classes used with the view. One included the panel for information from the consulting register, another was the panel to hold a new visit, and the final was a panel for the form to fill and submit. The form was designed to capture all of the unique data points for a new antenatal visit.

**Team 2 (Backend): Kyle Gendron, Aaron Harvey**

This team was tasked with building the AntenatalVisit object including its many fields and methods, as well as its large constructor. The team was also responsible for creating the controller responsible for adding information into the AntenatalVisit object from the UI. Instances of visits were then passed through a service class and stored to persistent flat file storage.

**Use Case List**

\* CHO conducts first antenatal exam for patient and records information into the register.

\* CHO conducts a follow-up exam and records new information into the register.

\* CHO searches for past antenatal exams of a patient.

\* CHO updates information for an exam.

\* CHO requests a Monthly Midwifes Report.

\* System alerts CHO when the next exam should be scheduled.

\* System alerts CHO when values are not within healthy ranges.

\* System alerts CHO when a patient has missed an appointment.

\* System prompts CHO to deliver critical information.

**Iteration One Use Case:**

CHO conducts first antenatal exam for patient and records information into the register.

1. CHO opens Consulting Register and records preliminary information.

2. Once complete the CHO opens the Antenatal Care Register.

3. Items up through and including Age are pulled and populated from the Consulting Register.

(Need to figure out how to make this happen)

4. CHO conducts exam and records data points above. (We may need to figure out how to save

progress in case power is lost).

5. CHO submits information.

6. System alerts CHO if items are missing.

7. CHO adds the missing information.

8. System alerts CHO that the register is complete and saved to storage.

**Tasks:**

\* Create a GUI with the required fields and checkboxes (est. 8 hours, act. 6.5 hours)

- create a tab for new visit (est. 1 hour, act. 30 mins)

- add labels for the items to be populated from the consulting register (est. 2 hours, act. 2 hours)

- add labels and text or check boxes for the unique antenatal exam items (est. 5 hours, act. 4 hours)

- create submit button that validates completeness and sends information to be stored (Validation moved to second iteration)

\* Controller class work (est. 8 hours, act. 5 hours)

- collect information from the GUI (est. 4 hours, act. 3 hours)

- use collected information to create a new visit instance

- send the visit instance to persistent storage

- if time allows, populate GUI with prior visit information

\* Build stub person object (est. 2 hours, act. 1 hour)

- create with already set fields

- blank constructor

- get and set methods

\* Create exam storage class (est 2 hours; act 2 hours)

- fields (We'll need one that references the Person object from the Consulting Register

- constructors

- methods: getters and setters

\* DAO/Storage (est. 3 hours, act. 2 hours)

- generate a DAO for storing the exam class

- generate a DAO for storing the DummyPerson class

\* Documentation (est. 2 hours, act. 1.5 hours)

- create basic JDoc comments

Accomplishments

* Created a working piece of software and the main skeleton of the program.
* Reached our goal of inserting a new antenatal care visit.
* Set up our documentation and wiki for use in the coming iterations.
* Generated a GUI that is expandable for functionality and usability in coming iterations.
* Team worked well collaboratively, assisting each other where necessary.

Issues

* Did not leave enough time at the end of the iteration for proper testing.
* Was unable to complete Java Docs for all classes.
* Unable to get to GUI validation – we will need to do this in the next iteration.
* Unable to use the actual person object while designing.
* Unable to refactor some of the controls in the GUI

Risks

* Incorporation of the real person object.
* Associating visits to the Person object.
* Validation and error checking.
* User experience