Use Cases:

- Display list of previous visits (Est: 10 hrs)
    - Basic flow:
        - User inputs patient identifier (still NHIS number or refactor to name?)
        - User is presented with Patient Information screen
            - Screen has list of previous visits
        - User selects desired visit
        - Details of desired visit are displayed on Visit screen
- Prepare monthly report for printing (Est: 4 hrs)
    - Basic flow:
        - User selects month and year from home screen
        - User clicks generate report
        - User is presented with formatted report ready for printing
    - Goal: get to easily readable screen

Refactoring

- UI cleanup (Josh & Dejan)
    - Visit page (3 hrs - enter visit information)
        - Split off hemoglobin into Pregnancy object
            - Add logic to ask for these values on the correct visits (initial and @36 weeks)
        -
    - Backend refactor for more front end functionality (? - previous visits)
    - More granular error checking (1 hr - enter visit information)
        - Be sure to indicate why a submit is failing (helper text)
        - Back end - define exceptions to tell us which is failing, update helper text
- Pregnancy model (Cam)
    - getActivePregnancy() return one object rather than list of objects (30 mins - enter visit information)
    - Verify that calculations will come from correct objects after refactoring pregnancy/visit objects
- Fold classes into attributes (Josh & Dejan)
    - PMCTC bools (15 mins)
    - Possibly shots if we don't implement functionality like alerts ?
- Logic for shot schedule ?
    - Eg if previously protected, should user have access to other shot fields?
    - If had 2nd shot, assume had 1st
    - Sickling positive has type, negative has no type

JUnit Testing

- Front-end testing?

Risks

- Working with incomplete Consulting Reg package
    - Need to get unique identifier that all patients in register will have
    - No UI
    - No add, edit, delete functionality
- Incorporate design patterns into code

Design Patterns

- Facade (~4 hrs)
    - Controller
        - Now: each individual controller implements their own view.
            - No reference to previous views
            - Solution: front facing UI controller which delegates to sub-controllers
        - Now: Too many dependencies throughout views and controllers. Eg visit view depends on patient service.
            - Solution: dependency injection via facade controller