Health Appointment Management System – Logbook

COMP1531 – S2 2018

Team Name: W13Bros

Team Members:

Jonathan Dufty	z5018512
Aran Pando	Z5160688
Martin Le	Z3466361
Natalie Nigro	z5229588
Stephen Kim	z5206897

Contents

Meeting 22/8

Present: Jon, Aran, Martin, Stephen

Items Discussed:

- Generate Epic Stories
    - Decided on 4-5 epic stories
    - Breaking down into user stories to be divided
- Logbook
    - To be done on google docs, minutes uploaded to git
- Everyone allocated 1-2 epic stories
- Classes
    - Some can be done before Use Cases/User story are finalised
    - Jon and Martin to develop basic classes
    - Everyones got allocated use cases to work ons

Next Meetings: Wednesday 29/08 – 3pm

- Finalise User Stories
- Elaborate on UML diagram
- Begin implantation planning

Meeting 29/8

Present: Jon, Martin, Stephen, Nat, Aran

Items:

1. User Stories
    - Need refining - everyone to critique others to get feedback
    - Need to be atomic, if there's an option to split it, then split it
    - Specific acceptance criteria, need to be able to complete a module, and once it ticks of the AC then it is done, some are really vague
    - Distinguish between AC and new features
    - Features that aren't necessarily in the brief explicitly…

    Added new user stories regarding login details, viewing profiles
    Everyone to go through and add comments to other user stories
    Add acceptance criteria for all
    Everyone to complete by Thursday/Friday
2. Classes
    - Start doing class diagrams
    - At least an early concept, have it done early to get feedback
3. Implementation
    - When do we want to start implementation (next week?)
    - Division of work…
4. Other notes
    - Repo organisation - classes in folders etc…?
    - Redefine User Points system? Make a bit smaller
    - Style convention - be consistent across everyones code

        module_name, package_name, ClassName, method_name, ExceptionName, function_name, GLOBAL_CONSTANT_NAME, global_var_name, instance_var_name, function_parameter_name, local_var_name (for example…. From stackexchange/PEP8)
5. Deadlines
    - User stories - Sunday
    - Class diagram - Wk 9?? - to be done well before
    - First release - Wk 9

Feedback 29/8

Feedback on user stories from laboratory session

- Need to be more specific with benefits. Some of them just restate a function, need to be a genuine benefit to the customer (almost like a selling point)
- Need to be a lot more specific with assessment criteria. Go into as much detail as possible
- Condense epic stories. Go from 8-> 3ish
- We had some features that might be nice, but aren't in the spec, so lets not create extra work for ourselves
- Group to go back and adjust user stories, condense into new epic stories
- Deadline by Friday to be reviewed by Saturday and submitted on Sunday

Meeting 5/9

Present: Jon, Aran, Martin, Stephen, Nat

1. CRC Cards
    - Basic drafting of CRC cards
    - Need to make sure class methods are for its own attributes
    - Discussed where to put searches -> within "manager" classes
    - Appointment Manager - universal list of appointments, indexed by ID
2. Class Diagram
    - Translation of CRC cards - Basic draft done by Jon/Martin
    - Need to finalise with CRC cards
    - Need to clear a few things up with tutors - getter/setter methods in class diagram
3. Implementation Plan
    - Need to find a way to get consistency across functions - will do this by specifying method outputs on UML
    - Options - split classes up per person, have to wait for functions to finish. Allow concurrent access to classes, add functions we need for

4. User Stories due for iteration 1

3.1 - Login Successfully (2 pts) – Responsible: Aran
Classes - User, User Manager, Patient, Provider

1.1 - Search for Centre (4 pts) – Responsible: Jon
Classes: Centre, CentreManager, SystemManager

1.2 - Search for Service (2 pts)
Classes: Provider, UserManager, SystemManager

1.3 - Search for providers (2 pts)
Classes: Provider, UserManager, SystemManager

1.4 - View provider profile (4 pts) – Responsible: Martin
Classes: Provider, SystemManager

1.5 - View center profile (4 pts)
Classes: Centre, SystemManager

1.6 - Book an appointment (4 pts) – Responsible: Nat
Classes: Patient, Provider, Centre, AppointmentManager
Responsible: Nat

2.2 - Patient view current appointments (4 pts) – Responsible: Stephen
Classes: Patient, AppointmentManager

2.6 - Provider view current appointments (4 pts)
Classes: Provider, AppointmentManager

5. Classes
User, user manager = Aran
Patient, provider = Nat
Centre, centre manager = Jon
Profiles = Martin, Appointments = Stephen

Meeting 10/09

Present: Aran, Nat, Jon

1. Finalise classes allocation
    - Completed skeleton for most classes, just need to write appointment class
    - Jon to do?? Stephen working on it already
2. Complete routes/templates
    - This will probably be the hard part to get used to
    - Python functions shouldn't take long, need to work on how to display to html
3. Unit tests
    - Everyone to do testing documents on their own branch
    - Using pytest framework
4. Directory management
    - How we want to organise files
    - Directories for model, routes, templates, static, tests etc…
    - Anything
    - Discuss in tute about how to implement

Discussion Points 11/09

Discussion Topics from outside of meeting

Issues/Roadblocks

- Centre profile Jinja error caused by a missing ' from <h3>Rating: {{ content['rating'] }}</h3> in the centre_profile.html template
- Availability attribute in provide is no longer dict{centre_name(str): hours(list<str>}, for it has been changed to {centre_id(int):{date(date):time_slots(list<str>)}}. This alternative means a booking can be made on a specified date
- Additionally, provider available time slots for a specific date is only added to availability after a booking is submitted to avoid making available slots for all dates in future.

Changes made/to make

- Persistence mechanism is through pickle. If no pickle file exists, take info from csv, using the bootstrap function from user manager and centre manager.
- Centre class to store provider objects, not just id, as it needs to access attributes
- Search implementation: Prefix match for all text searches, service search by drop down menu. Prefix match returns match is (A is in B) and A[0] == B[0].
    - Name search works for first name, last name or both
-

Meeting 12/09

Progress Report (Wednesday before Iteration 1):

User Story	Progress / comments	Approx User Points
3.1 – Login Successful	75% - framework done, need to test	1.5/2pts
1.1 – Search for centre	60% - functions complete + tested for all searches. Need to do front end	3/4pts
1.2 – Search for service		1/2pts
1.3 – Search for provider		1/2pts
1.6 – Book an appointment	20% - need to finalise appointment class	1/4 pts
1.4 – View provider profile	80% - Flask framework mostly done + HTML page	3/4 pts
1.5 – View centre profile	80% - Flask framework mostly done + HTML page	3/4 pts
2.2 – View current appointments (patient)	20% - Need to finish appointment class and patient/provider profile	1/4 pts
2.6 – View current appointments (provider)		1/4 pts

Other comments:

- Most people tracking well, should have app running and ready for testing by Friday morning
- Nat's away over the weekend (from Thursday) – rest of use to pick up user stories

Discussion Points 12/09 – 16/09

Discussion points outside of meeting, about issues encountered during the "cram weekend" before Iteration 1 deadline

Issues/Roadblocks

- Centre/Provider profiles – Issues with Jinja passing out direct classes.
    - Solution: Pass out string with centre_id/prov_email and use lookup function to return object
- Get_center_profile functions in system() use vars() method, but this maintains a reference, so deleting a field deletes it for the whole object
    - Solution: get_information() function in provider/centre/patient classes returns dictionary of relevant info to be displayed on profile for ease of use in Jinja
- CurrentUser – login isn't registering user so current_user just returns AnonymousUser
    - Solution: need to add @login_required decorator for all routes
- Need to implement "Add rating" feature as part of view profile user story
    - Jon to implement
    - Just as a form in centre/provider profile
- Persistence Issues – Outstanding Still
  Currently following lecture example of implementing pickle – e.g. run.py saves data after the app exits, and server.py loads data when the app starts. When the program exits, the versions of user_manager/centre_manager that run.py tries to save are un-updated so none of the new data is saved. Monitoring the vars during the app shows them being updated correctly, but as soon as CTRL+C is pressed, the vars revert back to zero. It appears that run.py takes a copy at the start and maintains that copy without getting updated values
    - Temporary Solution – Instead of saving data at the end of the app, save data after any update is made to user/centre managers (i.e. add rating or add appointment). Inefficient but it'll do for now until we get clarification
    - Note: The example code in the lectures has the same problem

Changes made/to make

- SystemManager removed as it was redundant. System.py still has helper functions such as function that links centres -> providers as both objects need to be intialised before this can be done
- Although appointment objects are stored in patient/providers now, appointment manager is kept as a global way of saving appointments
- Jon to cover view appointments, Aran to finalise booking with Stephen, Martin to finish profiles
- Search changed from separate URL to integrated search bar in nav bar – courtesy of martin

Iteration 1 Progress Check: 17/09

User Story	Progress / comments	Approx User Points
3.1 – Login Successful	100% - completed, tested, added logout + error message	2/2pts
1.1 – Search for centre	100% - functions complete + tested for all searches. Front end displays results. Integrated with nav bar	4/4pts
1.2 – Search for service		2/2pts
1.3 – Search for provider		2/2pts
1.6 – Book an appointment	100% - appointment class finished, time availability updates, and adds to appointment list	4/4 pts
1.4 – View provider profile	100% - Flask framework mostly done + HTML page. Web pages show basic information for providers and link to book, and rating option.	4/4 pts
1.5 – View centre profile		4/4 pts
2.2 – View current appointments (patient)	100%: List of appointments shows for both parties when appointment made. Links to centre/provider/patient all available from respective lists	4/4 pts
2.6 – View current appointments (provider)		4/4 pts
Additional User Stories		
2.4 – Give Rating	100% - implanted to finish profile pages	2/2 pts

Additional Comment/Notes for Iteration 2:

- Functionality completed for all required user stories. Still need to go back to improve error handling for system (not US specific as such).
- Tidy up webpage – currently no details were given about availability so could not test with different time availabilities
- Past appointments should be easy implementation – as we just monitor the 'past' Boolean
- May need to reassess User points for future iteration, not sure accurate our first guesses were

Changes for Iteration 2:

- Classes like appointment have a lot of getters/setters. Go through code and tidy up/ remove in not necessary
- Sort all python files into model/classes directory to tidy up repo. This wasn't done this iteration as it was still a manageable amount of files, and there would have been inconsistencies between file names between branches. Will do global change to master and have everyone merge to respective branches so all branches have same structure.

Velocity Chart: Iteration 1