
Meeting document that holds record of all meetings where all or nearly all members were present.

Does not include meetings of 3 group members or fewer. Includes meeting notes containing a summary of what was discussed/what was planned for future meetings.

Author: JT Kashuba, Noah Kruss, Logan Levitre, Zeke Petersen, River Veek

Group: TBD

Last Modified: 3/10/21

Initial Meeting – 2/12 (3:30pm-5pm) -- all members present

- Discussed project ideas, team name
- Established MoSCoW requirements
- Established tentative assignment breakdown and project timeline

Meeting – 2/15 (6:00pm-7:00pm) – all members present

- Stick with matrix view, do a tree view or something else later if we have time
- Landing page with full dropdown of everything in the DB (login/user viewing restrictions later)- decided that having admin/student buttons was beyond scope of this project
- Will need to stick with JSON data storage to make MongoDB happy – need to establish reasonable structure that we can work with going forward. Choosing this over python classes, because they can't be saved directly in Mongo and are less flexible, albeit easier to write.
- Display classes in a list in green, yellow, red depending on status as first thing they see—request the plan/forecast as a separate button/redirect.
- add course in progress vs add completed course options
- Dropdown for all possible classes to add (based on major requirements)

UPDATE ON 2/16 – We are going to shift gears into pickling the Student objects rather than saving in MongoDB—for our app, it will be more persistent and will allow for proper use of Python classes (rather than JSON)

Weekly Meeting – 2/17 (4:00pm-5:00pm) – all members present

- Figuring out how to handle users doing things wrong (let them realize it for when they add classes incorrectly and remove them)
- A piece of the user interface should include the unsaved changed (added or deleted files, perhaps as just a list of operations that were performed)
- Zeke walked through how pickling works
- Noah will start to make a presentation

- Presentation (content)
  - What the goal of our project is
    - ConOps of SRS
    - Note that we are initially designing just for UO CIS
  - Must haves, should haves, won't haves
  - This is what the interface will look like (demo?)
  - Diagram from SDS

explained in SRS sections 3.1, 3.2 (sort of) but typing here for "at a glance"

3 drop-down menus:
A list of all courses that can be added or removed (design choices still to be made)
Term (summer, fall, etc.)
Year (1st, 2nd, …, 5th) (for if they've already taken a course)

2 buttons: add course or remove class
Button for saving (storing in pickle file most likely?)
Button for displaying plan (redirect most likely?)

Weekly Meeting – 2/18 (6:00pm-7:00pm) – all members present
- Go over presentation
- Go over last-minute checks to project plan
- Clarified testing/development process and future scalability
- Begin with a static degree object to make sure all parts of the system are working—move to dynamic inputs later as time allows

Check-in Meeting – 2/21 (1:00pm- 1:35pm) – all members present
- Next task assignments:
  - JT – start writing installation instructions, create CIS degree object
  - Zeke and Logan – refine UI with Bootstrap
  - River – begin setting up a test suite for our class functions, pickling tests
  - Noah – generate forecasting function (to be later displayed in the UI)
- Questions for Juan?
  - Scope reasonable? Want more? Less?
  - Features that you think would be good to have? Most important features?
  - Are we on schedule?
  - Final presentation—what content are you expecting?

Flores said, "expecting a more complete development and documentation of the project. Static and dynamic diagrams to present use cases for situations we'll run into in the front and back end, storing things client side via pickling, etc."

Progress Meeting with Flores – 2/23 (10:30am-10:50am, 10:50-11:20) – all members present

- Explained design choices to Flores:
  - Decision to not use DB, and to instead use client-side storing of pickles
  - Decision to narrow scope to only account for CIS schedules
- Explained current degree objects to Flores
- Clarified with Flores that project scope was manageable
- Immediate follow-up meeting with JT, Noah, Logan, River:
  - Decided that scope should only encapsulate CIS major degree paths
  - Discussed how to handle electives (no concrete conclusion)
  - Discussed how to handle situations where user gets to choose multiple classes from set of options (no concrete conclusion)
  - Discussed how to handle multiple courses that share the same course number; decided to ignore course number (int) and to instead use course name (string)
  - Discussed adding a requirement type parameter to degree objects
  - Discussed adding difficulty level parameter (no concrete conclusion)

Weekly Meeting – 2/24 (4:00pm-5:00pm) – all members present

- Updates on frontend/backend
    - Still working on redirects, getting info from the HTML to the flask app
    - Working on logic for course forecasting

- Checklist for the frontend:
    - On select student, load correct student into app.py (or create a new object)
    - Create function- no input, output is list of all file names in pickle directory (for populating the initial dropdown) -- River
    - On save/save and display, send table info to be added to the loaded/created student in app.py
    - On display, properly redirect with forecast argument in the form that Jinja can handle

- Progress Meeting 2 w/ Prof Flores (questions/comments):
    - show what we have so far and ask for feedback on if there's something we're missing or if there's something that's unnecessary
    - Zeke will prep an architecture diagram

Check-in Meeting – 2/27 (2:30pm-3:30pm) – all members present

- Talked about adding UI help text (explain to user what does what), add ID number to the webpage so user know what entity they are manipulating
- Zeke will add framework for connecting the table info to flask so the backend can manipulate it
- Need to add a “create new student” option on the student select screen
- Talked about whether to add an option for the user to set the current term (decided against it) related to whether the user will be able to set a desired graduation date. Decided this is a “could have” rather than a “should have”
- Still nailing down design details for how to store course requirements (electives/core)
- Set benchmark/goal of having the scaffolding of a working system by end of the weekend. Reasoning for this is to have the next 2 weeks to connect back-end w/ front-end before working on system complexities and bells & whistles
- Came to an agreement on the system focusing specifically on being a Degree Planner for CIS Majors at the UO rather than a generalized Degree Planner for scope of this project submission. Decided we can continue working on the system as a side project for the group after submission to continue fleshing out a working system for all Major paths at the UO
- Courses in CIS_degree.py need to be updated to reflect which are available during Summer term
- Decided how to handle Gen Ed courses. Will create a Gen Ed Degree object to hold all gen ed courses, this design choice is working on the modularity for future versions of the system that will be able to handle ALL majors at the UO rather than only CIS majors

Progress Meeting with Flores – 3/2 (10:30am-10:50am) – all members present

- Talked about progress/roles—we are very close to done, most of the work left is connecting the front and back ends
- Asked about the graph structure of the pre_req calculations
- Commented on the appeal of the ability to show in the forecast tables if a course has been taken or is forecasted

Weekly Meeting – 3/3 (4:00pm-5:00pm) – all members present

- Chat about the syntax for the backend functions to use in app.py
- Create new function to generate list of courses (Noah)
- Finish function for parsing forecasted dictionary to format that the HTML can handle (JT)
- Discussed difficulty ratings—can add after the full system is working
- Tentatively set Saturday 3/6 at 11am for a meeting to discuss last commenting/SRS/SDS updates, etc.
- Tentatively set Monday 3/8 at 6pm for a presentation rehearsal

Impromptu Meeting – 3/4 (7:00pm-7:30pm) – all members present

- Brief demo— things work!
- Stopping and restarting the container works as expected and the pickles are being saved in the container rather than locally

- Will likely want to write some scripts for stopping and starting the container (so we don't have to ask the user to download Docker Desktop) – since they need to know how to run the ./run.sh script, we can expect them to know how to run start/stop scripts

Weekend Meeting – 3/6 (11:00am-11:50am) - all members present

SRS/SDS Documentation that needs updating:

- Interface, Logic, Pickling as modules
    - object definitions and plan logic (sub-dividing logic section into these 2 sub-modules)
- Flask as a module, how it directs traffic
    - specifically, how our index page is updating rather than redirecting
    - Needs a diagram
- Update architecture diagram to show the relation between modules as-is
- Extra diagrams for new modules (adding modules)
- Back-end needs to document the specifics that aren't handled by our system
    - Math placement test
    - Electives are handled, but in a certain way – explain
- "Add new student" + "add/remove course" exist, still need to add "save" + "save/display"
- Adding to "alternative designs" to show some direction the system could move towards for improving in the future
    - Display degree requirements and/or what still needs to be done on main user page
    - Needs to explain the system works on machines that can build Docker containers
    - Mobile does not support our system as of yet

Stuff that needs to be done or updated:

- Update and polish SRS/SDS
- Thorough testing
- Back-end
    - Course Difficulty (merge w/ pre-req logic to create "intelligent" plan)
    - Handle CIS 399 basically as a 399 elective
    - Same w/ 199
- User Instructions
- Installation Instructions
- README
    - Direct user/reader to other documents aka user instructions, installation instructions, etc. (brevity)
- Programmer Documentation (CREATE AND COMPLETE)
    - Includes nosetests explanation

- Presentation (meeting on Monday 3/8)
- Start/stop container shell scripts?
- User Documentation? (different from User Instructions)

Presentation Prep Meeting – 3/8 (6:00pm-7:00pm) - all members present

-Presentation assignments
- River – Introductions, schedule
- Logan – Problem statement, mental model
- JT – Future work, team org/task assignment
- Noah – Arch diagram, dynamic diagram
- Zeke – Demo, talk about mental model more during demo

- Talked about writing the programmer documentation (revisit Wednesday)
- Meeting 3/9 @ 7:30pm for presentation rehearsal
- Have everyone check off the SRS and SDS so we can convert to PDF tomorrow
- Finish other documentation, go over last minute rubric checks on Wednesday/Thursday

Final meeting – 3/10 (4:00pm-5:00pm) - all members present

- Going over Programmer Documentation, all hands on deck
- Add document headers to the non-code files
- Peer Evaluations need to get done
- Add project plan to repo
- Add excel spreadsheet to repo
- Add Agenda_MtgNotes to repo
- Test installation instructions
- Finalize README