**Team Members Period**

Ann Caplin 10

Jenny Chen

**Team Name**

JACC of all Trades

**Project Title**

Daily Planner: schedules your tasks for you

**I. Project Idea**

Organize a user’s day given information from an initial survey as well as what tasks they have to accomplish that day, how long they should take, and the time variability. The program is geared toward students. The organization of the schedule would be based on maximizing enjoyment & minimizing time spent working. The total time the schedule is predicted to take cannot be greater than the time frame the user has.

Examples: The software we’ve found just reminds the user of what they have to do that day and helps organize the user’s tasks. The interface the user puts information into will be similar to a school planner. The output will be a simple schedule.

Survey

Only taken once

Input: subject name, subject difficulty, enjoyment of subject (e.g., precalc, 80, 80)

Planner/task list

Inputted daily

Task list input: subject name + task name, expected time, time variability\*

Misc. input:

Breaks: required breaks, time range they could be in, how long that break would take (e.g, dinner, 5-7, 40 min)

Ideal stop time: The time the user would like to end their tasks at. This would normally be sleep time, but the user might also want to watch TV or something.

Time range: the time range these tasks should be completed in

\*How much the user thinks the expected time could realistically vary. Note that this time variation is based only on the task, not external conditions like the user’s energy. For example, I could know that I have precalc homework, but I’m not sure how many questions are going to be on it or how difficult it’s going to be. This would account for me inputting 40 minutes for time variability.

Schedule output

Computer outputs a text file that has a list of time range & subject name + task name for each task, listed in order of time.

**II. Critical Features**

1. Survey

Input text for subject name, subject difficulty, subject enjoyment each into different text boxes

Add a row of text boxes to fill out new information

Update information (writes to a text file)

2. Planner

Task list:

- Input information into a row of text boxes (subject name + task name, expected time, time variability)

- Add a row of text boxes

Breaks:

- Input info into a row, add another row

Ideal Time:

- Input info into a text box

Submit: button to get the output onto a schedule

3. Schedule

Simple text file to display the user’s ideal schedule with time ranges & tasks

**III. To be added later Features**

User can input initial energy into each planner (on a scale of 1-10, how energetic am I?)

Save different versions of the planner onto a text file, access different versions

Take user feedback about how the schedule worked

Modify survey (e.g., from a button in Planner GUI)

Display the schedule with GUI

Add user trait: Night Owl vs. Early Bird (affects energy over time)

Checkbox for each task: Necessary? That way, unnecessary tasks don’t have to be fully completed

User accounts

**IV. Development stages:**

**Version 1**

Ann & Jenny: Agree on formatting for survey text file, decide on schedule ranking system

Jenny:

- Basic survey GUI: finite # rows (no add button yet)

- “Update” button: writes to text file: make this method a boolean that returns true

Ann:

- Make basic framework: Objects needed w/their instance variables, methods (if any)

- Process survey text file

**Version 2**

Ann & Jenny: Agree on formatting for planner text file

Jenny:

- Add “add” button (increases the # rows)

- Implement “add” button in survey

Ann:

- Make equations for schedule (e.g., task time as a function of energy, difficulty, enjoyment, etc.)

- Scoring process for overall schedule

- Add support for breaks the user inputs, like dinner

- Process planner text file: convert everything to military time

- Run some examples to tweak constants/equations

**Version 3**

Jenny:

- Basic planner GUI: task list, break list, ideal stop time

Ann:

- Research different optimization algorithms (if not done already), choose one to use

- Use that to optimize the score of a schedule

**Version 4: Minimum functionality phase**

Jenny:

- Write planner GUI to text file: make this method a boolean that returns true

- Make sure all exceptions handled in GUI

- Make a method to make sure user inputs correct values

Ann:

- Make a program to call all these classes’ “run” methods (or a similar name, the point is that they are basically the main of that class): Survey run called, then Planner run, then OptimizeSchedule run (note that this run method is not in an infinite loop). These methods would be called once the text-writing boolean methods in Survey and then Planner return true.

- Debug a ton

**Version 5:**

Jenny:

- Make button to access & change survey

- Output schedule GUI

- Make any additional edits that there wasn’t time for in previous versions

Ann:

- After survey is changed, update the information for OptimizeSchedule: this will probably require waiting for the schedule with flawed survey information to be returned, then updating the information from the survey, and then making a new schedule with the updated survey information.

- Edit optimization algorithms

--------- Way too ambitious (only if we have time to add any extra features) ----------

**Version 6:**

Jenny:

- Allow user to input personality traits, like Night Owl

- Checklist if the task is necessary in Planner

Ann:

- Edit methods to handle these traits by changing energy over time if Night Owl & making a method to add unnecessary tasks at the end of everything else

**Version 7:**

Jenny:

- User feedback form: user inputs

-Personality and LifeStyle Survey

Ann

- Respond to user feedback about schedule, probably by changing certain constants for equations relating time & energy, enjoyment, etc.

**Version 8:**

Modifies Schedule as Go Along:

Changes schedule throughout the user’s day as they accomplish tasks on time or fail to do so

Jenny:

- Real-time schedule GUI: user inputs for each activity, whether or not task has been completed. When “completed” clicked, sends time information to Ann’s program to

Ann:

- Makes new schedule based off new time information

**Version 9:**

Allow User to Optimize Schedule Parts (Swapping Activities order, Adding Activities At Certain Slot)

Jenny:

- Schedule modification GUI

Ann:

- Calculate utility of this new schedule

**SURVEY**

|  |  |  |
| --- | --- | --- |
| **Subject Name** | **Difficulty (/100)** | **Enjoyment** |
| *Activity when done with tasks* | *0* | 100 |
| Precalc | 80 | 80 |



