Gujar Parth Shailesh
A0235354H
CVWO Mid-Assignment

Understanding what needs to be done

1. User profile
    1) Has continual access to the Internet
    2) User uses a desktop
    3) Needs a basic task management system
    4) Intuitive UI where you can drag tasks across categories
2. User requirements for the task management system
    1) Assigning priority to the task
        1) Let user add a deadline if required
        2) Let user assign a flag to mark the task as urgent if required
    2) Show urgent/pinned/overdue tasks first
    3) Information captured in each task
        1) Name of task ("Buy groceries")
        2) Date due – optional ("10 January 2022")
        3) Priority level – optional ("Urgent")
        4) Category – optional ("Shopping")
        5) Description – optional ("Eggs, milk")
        6) Has the task been completed yet ("No")
            1) If yes, the task should have its text struck through
    4) Intuitive UI
        1) Arrange tasks in columns (Default can be 3 columns, but user can change this – server should only return data that is sufficient to populate the desired number of columns)
            1) By date
                1) Tasks due today can be in the first column
                2) Tasks due tomorrow can be in the second column
            2) By category
            3) By name
            4) By priority level
            5) Advanced – by a custom order as defined by the user
        2) Overlay tasks on a calendar that shows the month or the week, so that the user can see immediately how many tasks are due on each upcoming day
        3) Button to delete each individual task, all completed tasks, or all tasks
        4) Button to enable user to view all completed tasks as well
        5) Advanced feature - Drag tasks across categories and auto-update the fields – for example, if a task about "Shopping" is dragged to the category about "Business," the tasks' category field should reflect "Business"
    5) Ability to create new categories as needed
        1) When creating a new task, the category inputted if not left blank should be added to the list of all categories
    6) Hide all completed tasks, but allow user to see them if needed
    7) Search button
    8) By default, user should see all the information about a task, but user can choose to see certain fields only, such as name of the task and deadline only
3. Use Cases
    1) An intuitive sequence of actions
        1) Sequence to add a new task
            1) Users enters information in a pre-existing empty form
            2) User clicks on a button to add the task

3) Server validates task
    1) Maximum characters allowed
    2) Type of information – for example, deadline should be in date format
4) Client device waits for the server's response

2) Sequence to change view
    1) User clicks on a button to change how the tasks are arranged on the screen
    2) Server returns the relevant tasks according to the view
3) Sequence to search and filter
    1) User uses the search button
    2) Server returns the relevant tasks
    3) The tasks on the screen are updated accordingly
4) Sequence to update/delete a task
    1) User clicks on the task
    2) User clicks on edit/delete button (with confirmation button to delete)
    3) Server returns result of operation

2) Account management
    1) Create user
    2) View user profile (email and password)
        1) Update profile
    3) Authenticate users (using JSON Web Tokens)
        1) Use 3rd party like GitHub or Google account
    4) Need a root/admin account as the base account
        1) Need the ability to add/remove privileges to perform CRUD for other users
        2) Ability to add/delete a user directly
    5) Delete user
3) Allow sessions
4) Cache data on numerous devices even if they are not connected to the Internet, so users can view their pending tasks
    1) Every CRUD operation to generate a token, and if the tokens on one device and the server are different, then it means that either the server or the device needs to be updated
        1) *Advanced feature – use Git to sync tasks instead of having a central server, but allow user to resolve merge conflicts*
    2) Block all CRUD operation if no access to the Internet
5) Search and filter
    1) By the fields as established above in 2.3)
6) Add all operations done to a log

Implementation details

Frontend: React.js
Backend: Go
Deployment: AWS Elastic Beanstalk to manage AWS EC2+S3/AWS Amplify

Sequence of steps

1. Frontend
    a. Signin page, signup page, home page with tasks, profile page, root user page
    b. Search and filter functionality
    c. Implement design (CSS)
2. Backend
    a. Set up server with MVC model

b. Set up API
c. Set up user account management
d. Deploy