RM LIFE PLANNER API DOCUMENTATION

**/api/users endpoints**

**key classes: UserAsParameter**

@router.post("/api/users/register")

Function: register\_user()

Parameters: user\_id: int, user: UserAsParameter

Description:

Register the given user and logs them in

Returns:

return {"user\_id": user\_id, "api\_key": UsersEndpoint.gen\_api\_key(user\_id)}, 200

Throws:

* 400 if username already exists
* 400 if username, password, or email do not meet the following standards:
  + username: < 24 characters
  + password: > 7 characters and < 32 characters
  + username and password: only contains alpha numeric characters and the following special characters: $, #, &, !, ?, @
  + email: is valid email

@router.get("/api/users/register/")

Function: is\_username\_in\_use()

Parameters: username: str

Description:

Checks if the username is already being used.

Returns:

return {"is\_in\_use": "True" if UsersEndpoint.users\_cursor.fetchone() is not None else "False"}, 200

Throws:

Nothing.

@router.post("/api/users/login")

Function: login()

Parameters: username: str, password: str

Description:

Logs in the user

Returns:

return {"user\_id": user\_id, "api\_key": UsersEndpoint.gen\_api\_key(user\_id)}, 200

Throws:

400 if username does not exist

401 if password is incorrect

@router.post("/api/users/logout")

Function: logout\_user

Parameters: user\_id: int, api:key: str

Description:

logs out the user and invalidates their api\_key

Returns:

return "User successfully logged out", 200

Throws:

401 if user\_id and api\_key do not match

@router.get("/api/users/")

Function: get\_user

Parameters: user\_id: int, api:key: str

Description:

returns the User object

Returns:

return {"message": "successfully got user", "user": json.dumps(user)}, 200

Throws:

401 if user\_id and api\_key do not match

@router.put("/api/users/")

Function: update\_user

Parameters: user\_id: int, api:key: str, updated\_user: UserAsParameter

Description:

updates the User object

Returns:

return "successfully updated", 200

Throws:

401 if user\_id and api\_key do not match or if the user attempts to change their username

@router.delete("/api/users/")

Function: delete\_user

Parameters: user\_id: int, api:key: str

Description:

updates the User object

Returns:

return "successfully deleted!", 200

Throws:

401 if user\_id and api\_key do not match

**/api/desires endpoints**

**key classes: DesireAsParameter**

These functions are mostly self-explanatory. They create, update, get, and delete desires. Desires are simply the general life desires a user has that he inputs into the application.

All functions check user authentication and throw 401 status code if they are not properly authenticated, or throw 400 if the specified function could not be completed due to a user error

@router.post("/api/desires")

def delete\_desire(user\_id: int, api\_key: str, desire\_id: int):

return {"message": "Desire created successfully", "desire\_id": GoalAchievingEndpoint.cursor.lastrowid}, 200

@router.get("/api/desires/{desire\_id}")

def get\_desire(user\_id: int, api\_key: str, desire\_id: int):

return {"message": "successfully got desire", "desire": json.dumps(desire\_dict)}, 200

@router.put("/api/desires/{desire\_id}")

def update\_desire(user\_id: int, api\_key: str, desire\_id: int, updated\_desire: DesireAsParameter):

return f"Desire with ID {desire\_id} updated successfully", 200

@router.delete("/api/desires/{desire\_id}")

def delete\_desire(user\_id: int, api\_key: str, desire\_id: int):

return f"Desire with ID {desire\_id} deleted successfully", 200

**/api/goals endpoints**

**key classes: GoalAsParameter**

These functions are mostly self-explanatory. They create, update, get, and delete goals. Goals are associated with 1 desire and are specific and measurable. They are often repetitive. Example goals could include “wake up on time every morning” or “get a job by the end of the month”.

All functions check user authentication and throw 401 status code if they are not properly authenticated, or throw 400 if the specified function could not be completed due to a user error.

@router.post("/api/goals")

def create\_goal(user\_id: int, api\_key: str, goal: GoalAsParameter):

return {"message": "Goal created successfully", "goal\_id": goal\_id}, 200

@router.get("/api/goals")

def get\_goal(user\_id: int, api\_key: str, goal\_id: int):

return {"goal: ": goal\_dict}, 200

@router.put("/api/goals")

def update\_goal(user\_id: int, api\_key:str, goal\_id: int, updated\_goal: GoalAsParameter):

return {"message": f"Goal with ID {goal\_id} updated successfully"}, 200

@router.delete("/api/goals")

def delete\_goal(user\_id: int, api\_key: str, goal\_id: int):

return {"message": f"Goal with ID {goal\_id} deleted successfully"}

**/api/plans endpoints**

**key classes: PlanAsParameter**

These functions are mostly self-explanatory. They create, update, get, and delete plans. Plans are associated with a goal and a CalendarEvent/Todo that, if completed, will help them meet their goal. Plans refer to a single planned event, so a goal such as “read a book every week”, would require at least 1 plan every week to read a book that week. When the user does their plan, an Action will be attached to the plan and CalendarEvent/Todo (see api/actions endpoints below). Only one action can be attached to a plan and only one plan can be attached to an action.

All functions check user authentication and throw 401 status code if they are not properly authenticated, or throw 400 if the specified function could not be completed due to a user error.

@router.post("/api/plans")

def create\_plan(user\_id: int, api\_key: str, plan: PlanAsParameter):

return {"message": "Plan created successfully", "plan\_id": plan\_id}, 200

@router.get("/api/plans/{plan\_id}")

def get\_plan(user\_id: int, api\_key: str, plan\_id: int):

return {"plan": res.\_\_str\_\_()}, 200

@router.put("/api/plans/{plan\_id}")

def update\_plan(user\_id: int, api\_key: str, plan\_id: int, updated\_plan: PlanAsParameter):

return {"message": f"Plan with ID {plan\_id} updated successfully"}, 200

@router.delete("/api/plans/{plan\_id}")

def delete\_plan(user\_id: int, api\_key:str, plan\_id: int):

return {"message": f"Plan with ID {plan\_id} deleted successfully"}

**/api/actions endpoints**

**key classes: ActionAsParameter**

These functions are mostly self-explanatory. They create, update, get, and delete actions. Actions are the things the user has done to meet their plans and goals. They measure exactly how much of the goal the user accomplished. They are attached to a CaledarEvent/Todo and a plan. Only one action can be attached to a plan and only one plan can be attached to an action.

All functions check user authentication and throw 401 status code if they are not properly authenticated, or throw 400 if the specified function could not be completed due to a user error.

@router.post("/api/actions")

def create\_action(user\_id: int, api\_key: str, action: ActionAsParameter):

return {"message": "Action created successfully", "action\_id": action\_id}, 200

@router.get("/api/actions/{action\_id}")

def get\_action(user\_id: int, api\_key: str, action\_id: int):

return res.\_\_str\_\_(), 200

@router.put("/api/actions/{action\_id}")

def update\_action(user\_id: int, api\_key: str, action\_id: int, updated\_action: ActionAsParameter):

return {"message": f"Action with ID {action\_id} updated successfully"}

@router.delete("/api/actions/{action\_id}")

def delete\_action(user\_id: int, api\_key: str, action\_id: int):

return "success", 200