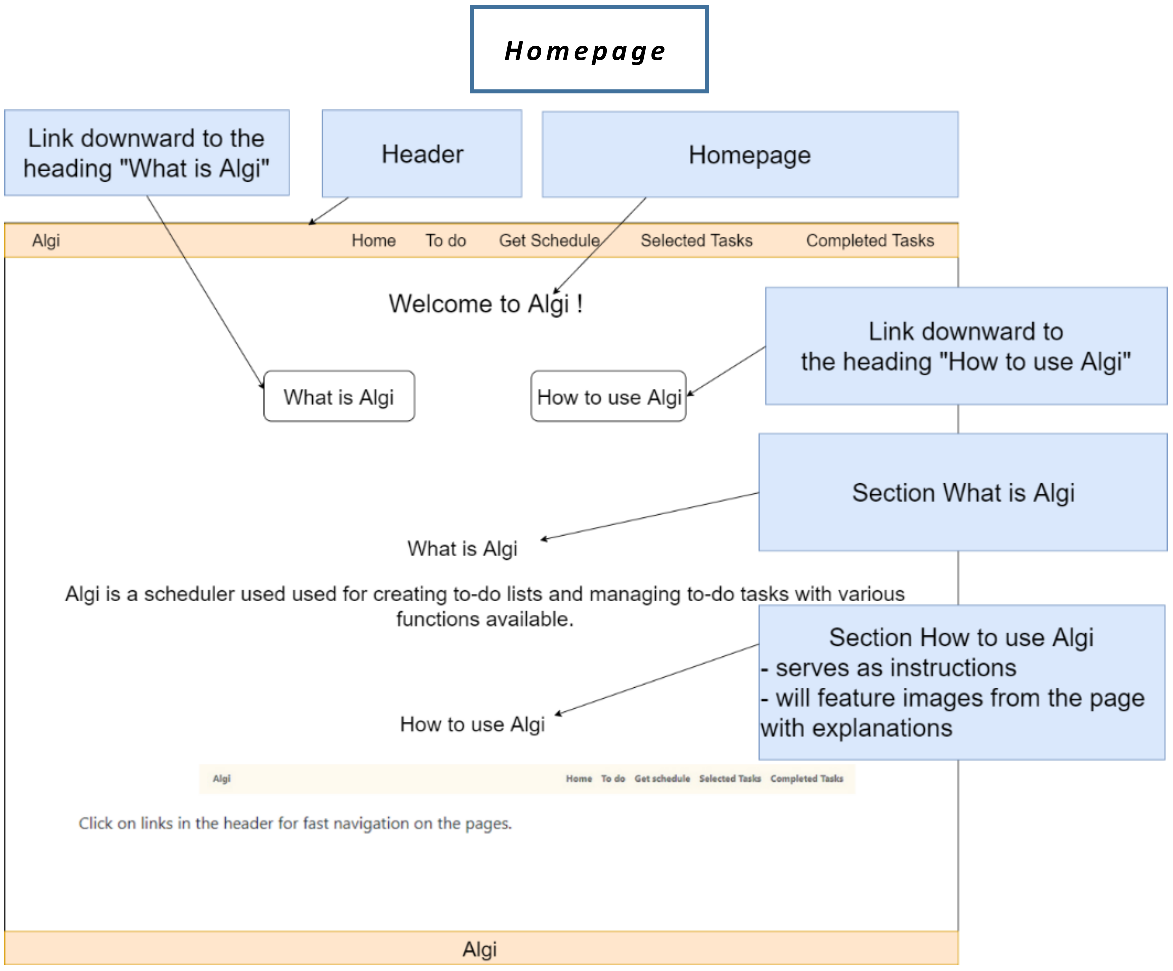
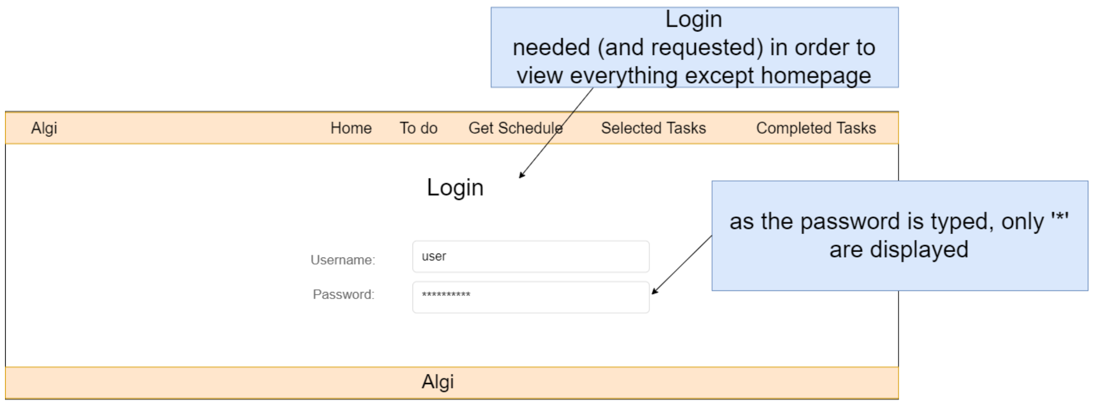
Criterion B: Design

Section I: Prototype solution


Homepage
Link downward to the heading "What is Algi"
Header
Homepage
Algi
Home
To do
Get Schedule
Selected Tasks
Completed Tasks
Welcome to Algi !
Link downward to the heading "How to use Algi"
What is Algi
How to use Algi
Section What is Algi
What is Algi
Algi is a scheduler used used for creating to-do lists and managing to-do tasks with various functions available.
Section How to use Algi - serves as instructions - will feature images from the page with explanations
How to use Algi
Algi
Home
To do
Get schedule
Selected Tasks
Completed Tasks
Click on links in the header for fast navigation on the pages.
Algi
Figure 1: *Design of the homepage*

Login

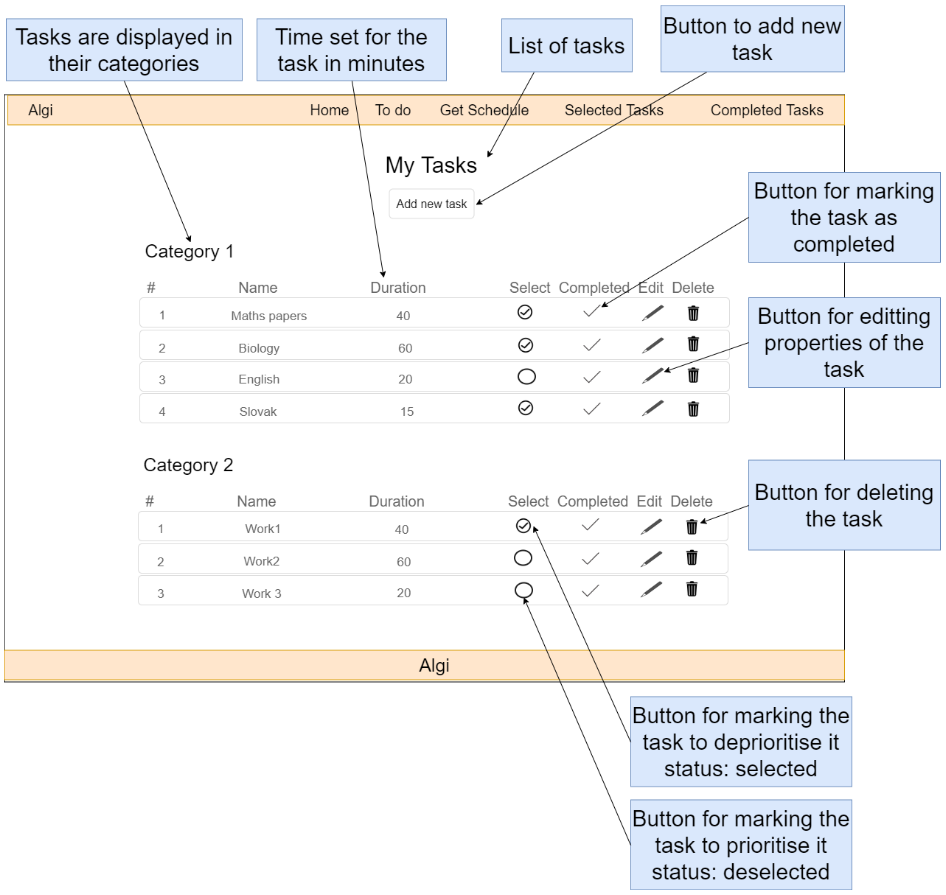

Login
needed (and requested) in order to view everything except homepage
Algi
Home
To do
Get Schedule
Selected Tasks
Completed Tasks
Login
as the password is typed, only '*' are displayed
Algi

Username: user

Password: **********

Figure 2: *Design of the Login page*

My Tasks


Tasks are displayed in their categories
Time set for the task in minutes
List of tasks
Button to add new task
Algi
Home
To do
Get Schedule
Selected Tasks
Completed Tasks
My Tasks
Add new task
Button for marking the task as completed
Button for editting properties of the task
Button for deleting the task
Algi
Button for marking the task to deprioritise it status: selected
Button for marking the task to prioritise it status: deselected

Category 1

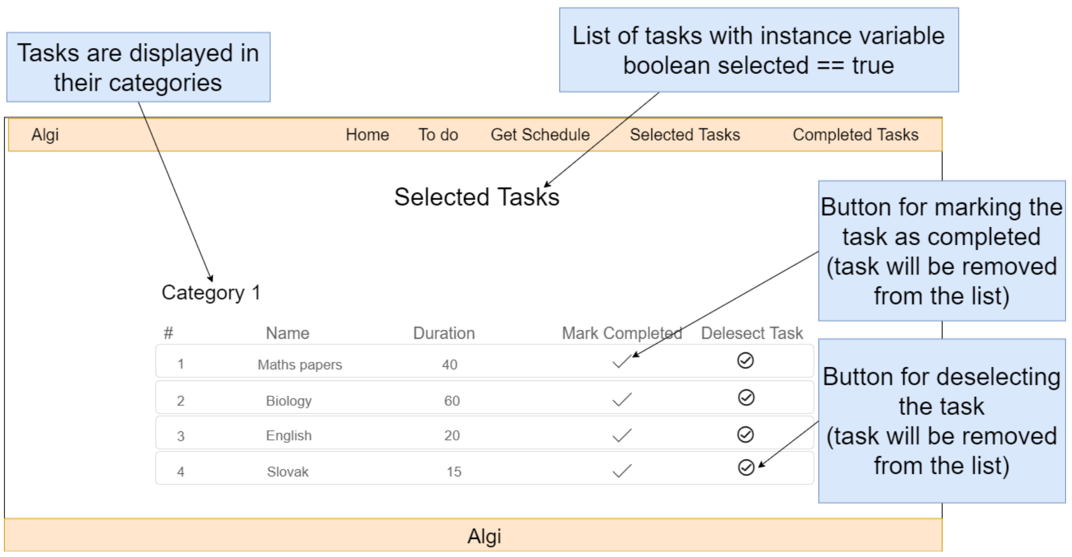
#	Name	Duration	Select	Completed	Edit	Delete
1	Maths papers	40	✓	✓	✎	🗑
2	Biology	60	✓	✓	✎	🗑
3	English	20	○	✓	✎	🗑
4	Slovak	15	✓	✓	✎	🗑

Category 2

#	Name	Duration	Select	Completed	Edit	Delete
1	Work1	40	✓	✓	✎	🗑
2	Work2	60	○	✓	✎	🗑
3	Work 3	20	○	✓	✎	🗑

Figure 3: Design of the My Tasks page

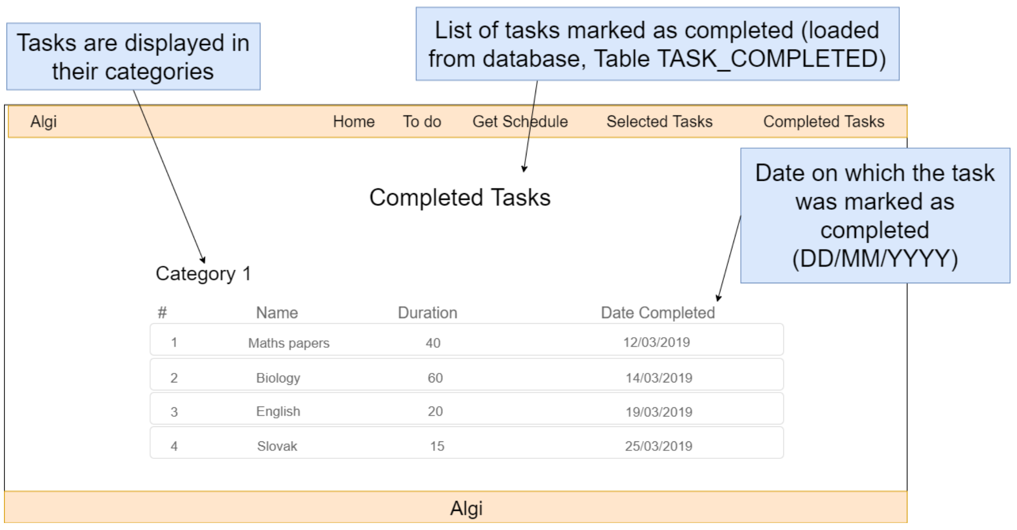
Selected Tasks


Tasks are displayed in their categories
List of tasks with instance variable boolean selected == true
Algi
Home
To do
Get Schedule
Selected Tasks
Completed Tasks
Selected Tasks
Button for marking the task as completed (task will be removed from the list)
Category 1
Button for deselecting the task (task will be removed from the list)
Algi

#	Name	Duration	Mark Completed	Delesect Task
1	Maths papers	40	✓	⊘
2	Biology	60	✓	⊘
3	English	20	✓	⊘
4	Slovak	15	✓	⊘

Figure 4: Design of the Selected Tasks page

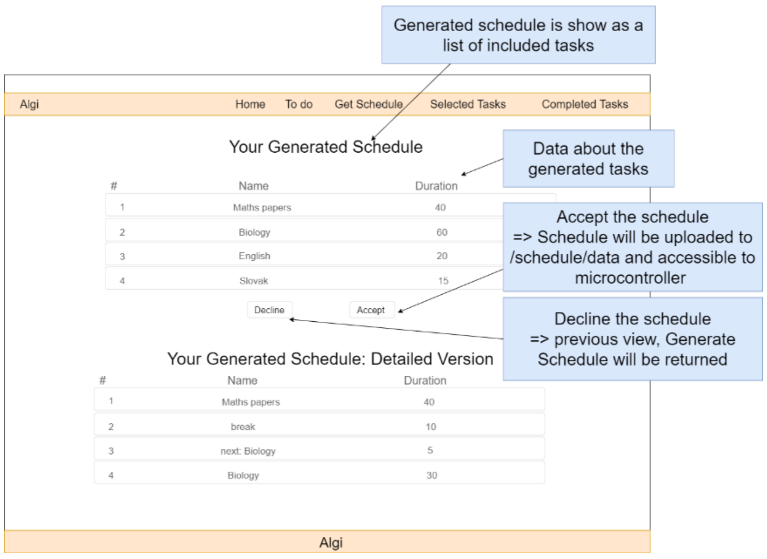
Completed Tasks


Tasks are displayed in their categories
List of tasks marked as completed (loaded from database, Table TASK_COMPLETED)
Algi
Home
To do
Get Schedule
Selected Tasks
Completed Tasks
Date on which the task was marked as completed (DD/MM/YYYY)
Completed Tasks
Category 1
Algi

#	Name	Duration	Date Completed
1	Maths papers	40	12/03/2019
2	Biology	60	14/03/2019
3	English	20	19/03/2019
4	Slovak	15	25/03/2019

Figure 5: Design of the Completed Tasks page

Your Generated Schedule



Generated schedule is show as a list of included tasks

Your Generated Schedule

#	Name	Duration
1	Maths papers	40
2	Biology	60
3	English	20
4	Slovak	15

Data about the generated tasks

Accept the schedule
=> Schedule will be uploaded to /schedule/data and accessible to microcontroller

Decline the schedule
=> previous view, Generate Schedule will be returned

Your Generated Schedule: Detailed Version

#	Name	Duration
1	Maths papers	40
2	break	10
3	next: Biology	5
4	Biology	30

Figure 6: Design of the Your Generated Schedule page

Section II: Data

Strored data will include information about:

- tasks
- user login credentials
- uploaded tasks
- completed tasks

One table in database for each group of data will be used.

H2 database functions for work with data	
findAll()	Returns all objects from the chosen table
MyUser findUserByUsernameAndEncryptedPassword(String userName, String password)	Returns all users that match the passed parameters
save()	Saves object to the database
getOne(Long id)	Returns object from database with the same id
delete(Task task)	Deletes the task from database

Section III: UML diagrams

List of classes	
Name of the class	**Purpose of the class**
ScheduleController	Handling communication between client (web browser) and server for schedule-related pages
TaskController	Handling communication between client (web browser) and server for task-related pages
UserController	Handling communication between client (web browser) and server for user-related pages
ScheduleFormDTO	Managing data inputted by the user for the generated schedule
TaskFormDataDTO	Managing data inputted by the user about the tasks
UploadedDTO	Managing the to-be-uploaded tasks
CredentialsDTO	Managing user login information
CompletedTask	Handling data about completed tasks
MyUser	Managing user data
Task	For manging tasks
Manager	Handling task categories
ScheduleService	Handling of processing of schedule-related data
TaskService	Handling of processing of task-related data
UserService	Handling of processing of user-related data
Validation	Methods for validating of inputs

Package: controller

class ScheduleController		
- ScheduleService scheduleService		
- TaskService taskService		
Get	+ String loadTasks(Model model)	Encloses a list (List<Task>) of all tasks requested from taskService class and list (List<String>) of categories to the Model

Get	+ List<UploadedDTO> uploadData()	Encloses a list of tasks marked as to be uploaded using scheduleService to the Model
Get	+ String getSchedule(Model model)	Request the view of generateSchedule.html
Post	+ String generateScheduleButtons(@RequestParam(value = "action") String action)	Request the scheduleService to handle actions call by the buttons for generating a schedule
Post	+ String generateSchedule(@ModelAttribute ScheduleFormDTO scheduleFormDTO, Model model)	Calls the scheduleService to generate a schedule and calls requests the view with the generated schedule
Get	+ String displaySchedule(@ModelAttribute List<Task> display, Model model)	Request the view of the schedule with the loaded Tasks using the TaskService

class TaskController		
- TaskService taskService		
Get	+ String newTask(Model model)	Request the view of new task form
Post	+ String createTask(@ModelAttribute TaskFormDataDTO taskFormDataDTO)	Call taskService to create a new task from the variables passed as parameters from the view
Get	+ String editTask(@PathVariable("id") Long id, Model model)	Request the view of task editting form
Post	+ String editTask(@ModelAttribute Task task)	Call TaskService to save the changed parameters of a task
Get	+ String displayCompleted(Model model)	Get from TaskService all tasks marked completed and call view to display them
Post	+ String edittingTask(@PathVariable("taskIdParameter") Long id, @RequestParam(value = "action") String functionToPerform)	Call TaskService on a task to performed the inputed function functionToPerform
Get	+ String loadSelected(Model model)	Get tasks marked as selected using TaskService and call view to display them
Post	+ String changeSelected(@PathVariable("taskIdParameter") Long id, @RequestParam(value = "action") String functionToPerform)	Calls TaskService to perform functionToPerform

class UserController		
- UserService userService		
- UserRepository userRepository		
Get	+ String loginView()	Request the view of login form
Get	+ String viewHome()	Request the view of homepage
Get	+ String blank()	Request the view of homepage

Get	+ String changeLoginView(Model model)	Call the view of form for changing login credentials
Post	changeLoginCredentials(@ModelAttribute CredentialsDTO credentialsDTO, Model model)	Call UserService to validate and process the inputted data, request changeLogin from the view
Get	changedLoginView()	Call the view of loginChanged

Package dto

class CredentialsDTO
- String oldUsername
- String newUsername
- String oldPassword
- String newPassword
+ CredentialsDTO()
+ CredentialsDTO(String oldUsername, String newUsername, String oldPassword, String newPassword)
getters
setters

class ScheduleFormDTO
- String sessionLength
- String breakLength
- String breakFrequency
- String categorySelected
+ ScheduleFormDTO ()
getters
setters

class TaskFormDTO
- String name
- String duration
- String taskCategory

getters
setters

class UploadedDTO
- long id
- String name
- Integer duration
+ UploadedDTO()
+ UploadedDTO(String name, Integer duration)
getters
setters

Package entity

class Task
- long id
- String name
- int duration
- String taskCategory
- boolean selected
- boolean forUpload
+ Task ()
+ Task (String name, int duration, String taskCategory, boolean select)
getters
setters

class CompletedTask
- long id
- String name
- int duration
- String taskCategory

- LocalDate localDate	
- String localDateString	
+ CompletedTask ()	
+ CompletedTask (Task task)	
getters	
setters	
+ void setTodayAsDate()	Sets the localDate and localDateString of this task to today

class MyUser
- String username
- String encryptedPassword
- String password
+ MyUser ()
+ MyUser (String username, String encryptedPassword, String password)
getters
setters

Package model

class Manager
- List<String> categoriesL
- int numberOfCategories
+ Manager ()
getters
setters

Package service

class ScheduleService
- TaskRepository taskRepository

- UploadedDataRepository uploadedDataRepository	
- TaskService taskService	
+ List<Task> generateSchedule(int sessionLength, String categorySelected)	generate a list of tasks based on the input parameters of length of the planned session and category of the tasks featured in the session
+ List<UploadedDTO> getDataForUpload(List<Task> display, int breakFrequency, int breakLength)	adds additional details, such as breaks and time to finish and prepare for the next task, to a list carrying all tasks featured in the generated schedule, uploads and return the list with all instructions (breaks and tasks)
+ Integer getTotalDuration(List<UploadedDTO> list)	returns durations of all tasks in the list passed as a parameter of the function
+ void buttonsGenerateSchedule(String functionToPerform)	performs an action of declining or accepting a schedule based on the passed parameter
+ void saveUploadDataToRepository(List<UploadedDTO> list)	saves all tasks in the list passed as a parameter to the database
+ List<UploadedDTO> getUploadDataFromRepository()	returns all objects from the database, Table UPLOADEDDTO
- clearUploadDataRepository()	clears the whole table UPLOADEDDTO in the database
- static List<Task> getCategoryTasks(String category, List<Task> tasks)	returns all task of the category String category (first parameter) in the list List<Task> tasks
- static Task[] sortTasks(List<Task> tasks)	sorts the list List<Task> tasks
- static int partition(Task arr[], int low, int high)	partition function needed by the QuickSort sorting algorithm
- static void sort(Task arr[], int low, int high)	QuickSort sorting algorithm

class TaskService	
- TaskRepository taskRepository	
- CompletedTasksRepository completedTasksRepository	
- UserRepository userRepository	
+ List<Task> getAllTasks()	returns all tasks from the database
+ boolean createTask(String name, String duration, String taskCategory)	creates a new task from the parameters and saves it to the database
+ Task viewEditTask(Long id)	finds a task in the database by its id
+ void saveEditTask(Task task)	updates an existing task in the database
+ List<Task> getSelectedTaskList()	returns selectedTasksList
+ void edittingTask(Long id, String function)	performs deletion on a task from database or marks it as completed or selected based on the String function parameter value
+ void changeSelected(Long id, String function)	updates a status of a task in the database as selected or completed based on the String function parameter value

+ List<Task> loadSelected()	returns all tasks with instance variable selected == true
+ List<CompletedTask> listCompleted()	returns all tasks from the database Table COMPLETED_TASK
+ void setTasksForUploadFalse()	sets all the instance variable boolean forUpload of all tasks in the database to false
+ void setTasksForUpload(List<Task> tasksForUpload)	sets all the instance variable boolean forUpload of tasks in the list tasksForUpload to true and saves them to the database table UPLOADEDDTO
+ static List<Task> getTasksForUpload(List<Task> tasks)	returns all tasks from database table UPLOADEDDTO

class UserService	
- UserRepository userRepository	
+ boolean changeLoginCredentials(String oldUsername, String newUsername, String oldPassword, String newPassword)	returns true if the original login credentials were successfully changed to the new ones
+ void registerUser(String username, String password)	creates a new user with String username and String password and saves the user to the database
+ boolean authentificateUser(String userName, String password)	checks whether the user with username String userName and password String password exists

class Validation	
+ boolean validateInputInt(String input)	validates an input, whether it is an integer
+ boolean validateString(String input)	validates an input, whether it contains some characters

class IAssesment1Application	
+ static void main(String[] args)	main

List of html files
changeLogin.html
completed.html
editTask.html
footer.html
generateSchedule.html
generatedSchedule.html
header.html
home.html
login.html

loginChanges.html
new_task.html
schedule.html
selected.html
signup.html

Section IV: Program flowcharts

Client / Server model

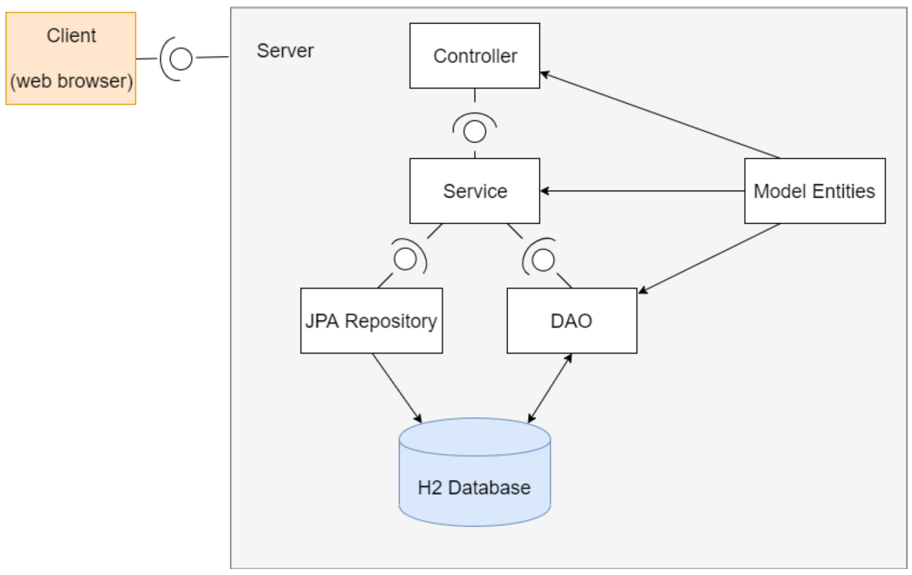
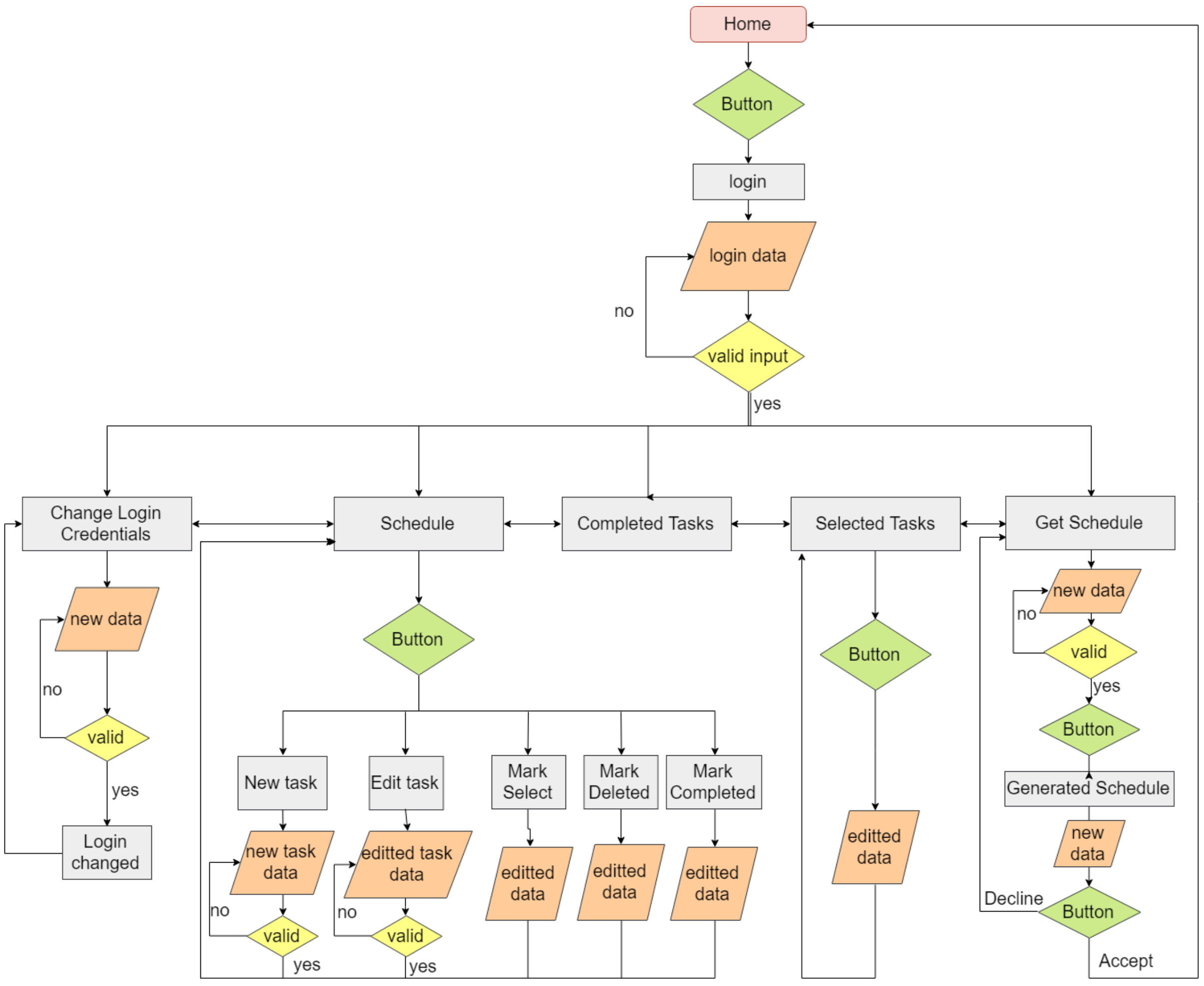


Figure: Client / Server model of the application showing the general structure of the project

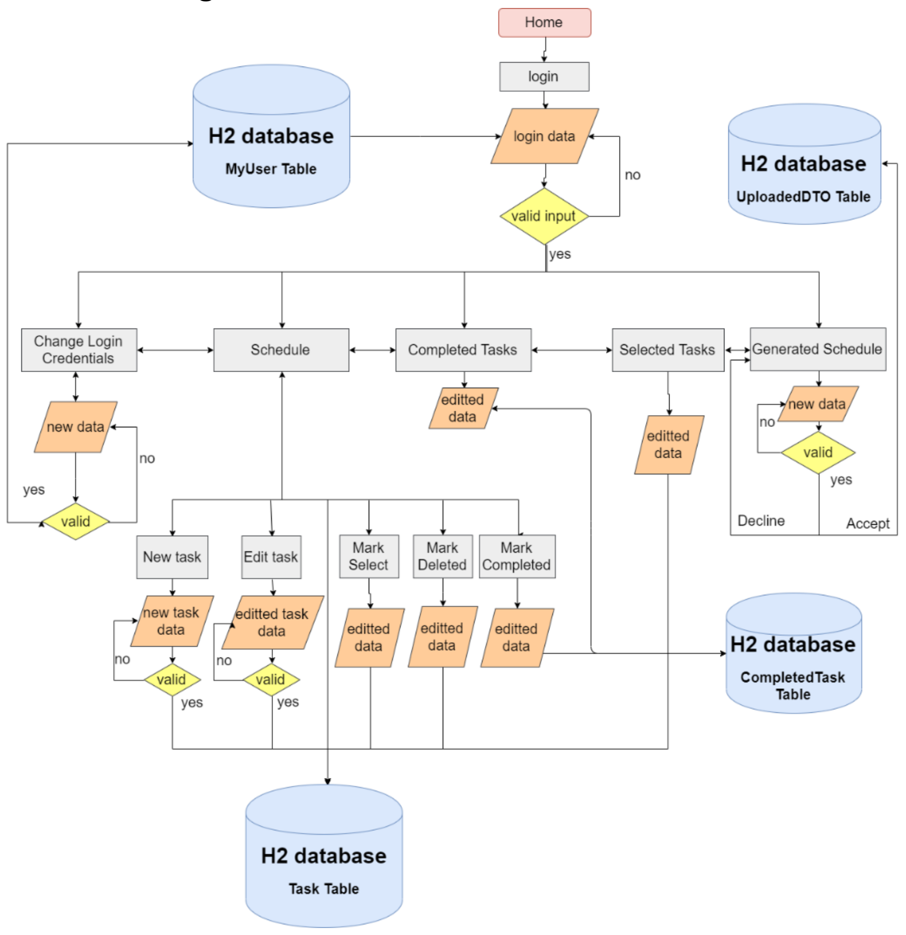
In the diagram above, DAO represents a Data Access Object, a pattern to persist domain objects into the H2 database, implemented as a class with methods concerning a particular domain entity type (Thinking in objects, 2017).

JPA (Java Persistence API) Repository is used for persisting Java objects into relational databases (Pollack et al, 2012).

General program flowchart



Program flowchart with a focus on data flow



Section V: Code

1. Algorithm for generating schedule

The purpose of the algorithm will be generating a schedule - list of successive tasks and breaks. The schedule will be generated based on data about the planned learning session inputted be the client, which will be:

- SESSIONLENGTH
  - length of the planned session
- SELECTEDCATEGORY
  - category of the tasks featured in the session

and data regarding the tasks themselves:

- TASKNAME
  - name of a task

- TASKDURATION
    - time the client allocated to a specific task
- TASKCATEGORY
    - category of a task

In the session, tasks of various length should alternate (so that the client can remain concentrated). The algorithm will output a list LIST of all chosen tasks, which altogether have a length of smaller than the SESSIONLENGTH.
Various functions, as for example for sorting the tasks from SELECTEDCATEGORY in the ascending order, or `getCategoryTasks` to get all task with set SELECTEDCATEGORY, the will be needed.

```
CATEGORYTASKS = getCategoryTasks(SELECTEDCATEGORY, ALLTASKS)
DURATIONSTOTAL = getDurations(CATEGORYTASKS)
DURATION = 0, I = 0, LENGTH = CATEGORYTASKS.length

if DURATIONSTOTAL < SESSIONLENGTH then
        LIST = CATEGORYTASKS
else
        CATEGORYTASKS = sortTasks(CATEGORYTASKS)
        loop while DURATION = CATEGORYTASKS.get(I).getDuration() <= SESSIONLENGTH
         AND ( I < LENGTH/2  OR  (I==LENGTH/2  AND LENGTH%2 = 1 ) )

                LIST.add(CATEGORYTASKS.get(I))
                DURATION = DURATION + CATEGORYTASKS.get(I).getDuration()
                BACK = LENGTH-I-1;

                if DURATION + CATEGORYTASKS.get(BACK).getDuration() <= SESSIONLENGTH
                AND I<LENGTH/2
                        LIST.add(CATEGORYTASKS.get(BACK))
                        DURATION = DURATION + CATEGORYTASKS.get(BACK).getDuration()
                end if

                I = I+1
        end loop
end if
```

If the duration of all task present in the category (`DURATIONSTOTAL`) is smaller than the `SESSIONLENGTH`, all tasks are added to the `LIST`.
Otherwise, a while loop is entered. In each itineration in the while loop, Ith task from the front (smallest) and Ith task from the back (largest) in the `CATEGORYTASKS` will be checked.
The while loop end when all tasks are checked (number of iterations is equal to the half of the even number of tasks, or half+1 iterations in case of uneven number of tasks) or when there is not enough remaining time for other task to be added (`DURATION = CATEGORYTASKS.get(I).getDuration() <= SESSIONLENGTH`).

2. H2 Database

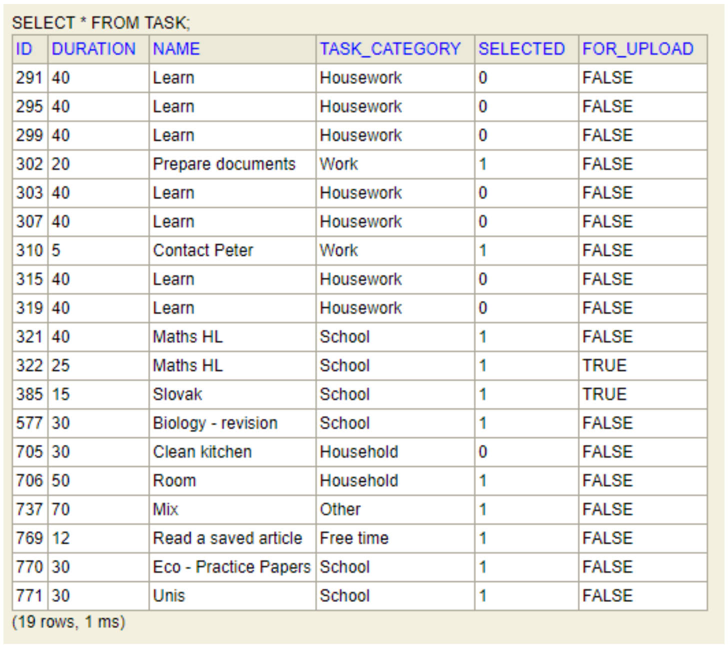
For storage of data about the tasks and users, an open-source Java database called H2 database will be used. The reasons for the choice of particularly this database were:

- written in Java – uses a programming language already uses in the application

- provides a browser-based console application: (H2 Database Engine)
    - console view can be used to display created tables with all the data they hold (Dashora, 2019)
- can be configured so that the tables will be persistent
- small footprint: ~ 2 MB jar file size (H2 Database Engine)

TASK table

Column	Data type	Description
ID	Long	ID of the task, used as a primary key to uniquely identify a row in the table, autogenerated
duration	Integer	duration of a task
name	String	name of a task
taskCategory	String	task category of a task
Selected	Boolean	selected denotes whether a task was marked by the user to be done as next, the soonest possible
forUpload	Boolean	forUpload denotes whether a task was marked by the user to be uploaded to the device / be upload if upload is requested

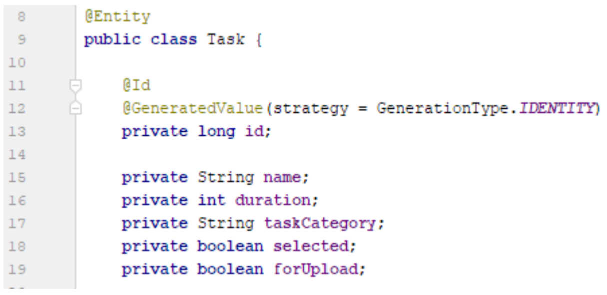


SELECT * FROM TASK;

ID	DURATION	NAME	TASK_CATEGORY	SELECTED	FOR_UPLOAD
291	40	Learn	Housework	0	FALSE
295	40	Learn	Housework	0	FALSE
299	40	Learn	Housework	0	FALSE
302	20	Prepare documents	Work	1	FALSE
303	40	Learn	Housework	0	FALSE
307	40	Learn	Housework	0	FALSE
310	5	Contact Peter	Work	1	FALSE
315	40	Learn	Housework	0	FALSE
319	40	Learn	Housework	0	FALSE
321	40	Maths HL	School	1	FALSE
322	25	Maths HL	School	1	TRUE
385	15	Slovak	School	1	TRUE
577	30	Biology - revision	School	1	FALSE
705	30	Clean kitchen	Household	0	FALSE
706	50	Room	Household	1	FALSE
737	70	Mix	Other	1	FALSE
769	12	Read a saved article	Free time	1	FALSE
770	30	Eco - Practice Papers	School	1	FALSE
771	30	Unis	School	1	FALSE

(19 rows, 1 ms)

Figure: H2 Console view of the TASK table

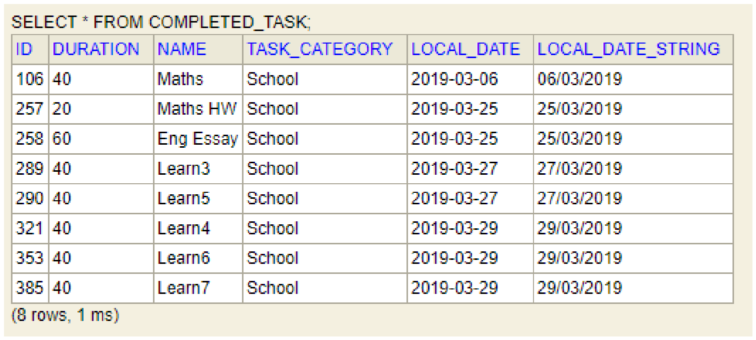


```
@Entity
public class Task {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    private String name;
    private int duration;
    private String taskCategory;
    private boolean selected;
    private boolean forUpload;
```

Figure: Code from class Task

COMPLETED_TASK table

Column	Data type	Description
ID	Long	ID of the task, used as a primary key to uniquely identify a row in the table, autogenerated
duration	Integer	duration of a task
name	String	name of a task
taskCategory	String	task category of a task
localDate	LocalDate	date on the task having been marked as completed
localDateString	String	date on the task having been marked as completed in the String form



SELECT * FROM COMPLETED_TASK;

ID	DURATION	NAME	TASK_CATEGORY	LOCAL_DATE	LOCAL_DATE_STRING
106	40	Maths	School	2019-03-06	06/03/2019
257	20	Maths HW	School	2019-03-25	25/03/2019
258	60	Eng Essay	School	2019-03-25	25/03/2019
289	40	Learn3	School	2019-03-27	27/03/2019
290	40	Learn5	School	2019-03-27	27/03/2019
321	40	Learn4	School	2019-03-29	29/03/2019
353	40	Learn6	School	2019-03-29	29/03/2019
385	40	Learn7	School	2019-03-29	29/03/2019

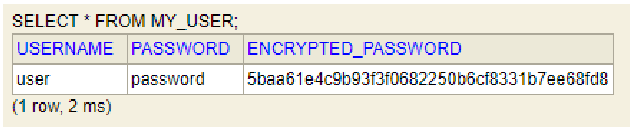
(8 rows, 1 ms)

Figure: H2 Console view of the COMPLETED_TASK table

MY_USER table

Column	Data type	Description

username	String	username serves for both user identification and as an ID of a user, used as a primary key to uniquely identify a row in the table
encryptedPassword	String	users's password hashed with Secure Hash Algorithm 1 hash function
password	String	users's password



SELECT * FROM MY_USER;

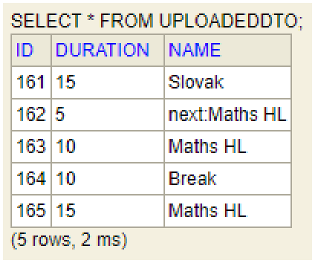
USERNAME	PASSWORD	ENCRYPTED_PASSWORD
user	password	5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8

(1 row, 2 ms)

Figure: H2 Console view of the MY_USER table

UPLOADEDDTO table

Column	Data type	Description
ID	Long	ID of the task, used as a primary key to uniquely identify a row in the table, autogenerated
duration	Integer	duration of a task
name	String	name of a task



SELECT * FROM UPLOADEDDTO;

ID	DURATION	NAME
161	15	Slovak
162	5	next:Maths HL
163	10	Maths HL
164	10	Break
165	15	Maths HL

(5 rows, 2 ms)

Figure: H2 Console view of the UPLOADEDDTO table

Test plan for success criteria

# SC	Success criterion	Test plan
1	Client can manage tasks • add a new task • modify existing tasks • delete tasks	All tasks can be displayed → creation of a new task → edit a task → delete a task

1	Add a new task	Create a new task with a wrong input form (not integer for Integer duration)
2	List of tasks for upload is accessible and in an appropriate form	Display all tasks for upload
3	List of tasks for upload is always up-to-date	Mark a task for the upload + display all tasks for upload
4	Dates of completion of the tasks can be accessed	Display a list of all completed tasks
5	A schedule proposal can be constructed based on the inputted tasks to do	Generate a schedule using Generate Schedule
5	A schedule proposal can be constructed based on the inputted tasks to do	Generate a schedule using Generate Schedule with session length of 10 minutes
5	A schedule proposal can be constructed based on the inputted tasks to do	Generate a schedule using Generate Schedule with wrong inputs
6	Instructions for how to use the application are available	Click on How to use Algi link on homepage
7	Application can be easily navigated	Use the header to go from one page to another
8	Restricted access to data	Inability to delete a task from the To do list
8	Restricted access to data	Put wrong inputs into the login form

Bibliography

Dashora, (2019) S. Setting up H2 database with Spring Boot. Retrieved from http://progressivecoder.com/2019/01/setting-up-h2-database-with-spring-boot/

H2 Database Engine. Retrieved from http://www.h2database.com/html/main.html

Pollack, M., Gierke, O., Risberg, T., Brisbin, J., Hunger, M. Spring Data: Modern Data Access for Enterprise Java (2012) United States: O'Reilly Media

Thinking in Objects. (2012) Retrieved from https://thinkinginobjects.com/2012/08/26/dont-use-dao-use-repository/