Web Applications A.Y. 2022-2023

Homework 1 – Server-side Design and Development

Master Degree in Computer Engineering
Master Degree in Cybersecurity
Master Degree in ICT for Internet and Multimedia

Deadline: 28 April, 2023

Group AmuParkWA	Project Amusement Park Web App	
Last Name	First Name	Student Number
Huimin	Chen	2071518

Objectives of the System

The objective of the project is to develop website named WorkFlix for project management. The project is mainly focused on aspects related to the exchange of the data between users and the back end.

Main Functionalities

The web application WorkFlix enables users to organize projects and everything related to them into boards. With WorkFlix, users can find all kinds of information regarding team collaboration and project management. This application has several key features. Here is a workflow of how to use this web application as different roles.

- Sign-up/Log in: available to unregistered users and registered users.
  - Sign-up:
  - Log-in:
- Create a WorkSpace in WorkFlix
  - Create new WorkSpace
  - Edit the WorkSpace
- Create a Board in WorkFlix
  - Create new board
  - Name the board
  - inserting new measurement files: such files are uploaded via web interface and processed offline.
- Create Sub-board in WorkFlix
  - insert new parks and edit existing ones. to be used when rides have not been deployed before on such park
  - insert new models and edit existing ones: used when the fictitious builder develops a new model
  - insert new rides and edit existing ones.

Roles can be sorted according to the privileges they grant. In particular, admin > manager > editor > users. This means that a user can access all the areas of the web-site granted by their and lower roles.

The site has an accessory area which allows every user to either register or login to the application. Note that, if the registration is completed successfully the user should be notified via email.

Presentation Logic Layer

The website will be divided into the following pages:

- Landing page: The main purpose of the landing page is to encourage visitors to take action including log in or sign up for WrokFlix website.
- Board page: it is created under a workspace to organize tasks. On a board page, users can keep track of information about projects and workflows, which helps you collaborate with your colleagues.
- Login page: allows the users login to the web application.
- Sign up page: allows the users to register to the web application.

Landing Page

The landing page is divided into two sections: "header" and "body".
In the 'Header' area, there are two sub-sections. On the left, the logo of our website 'WorkFlix' is displayed. In the second homework, our team will design the log. On the right, the main navigation is composed of two major functions of the landing page, namely, log-in and sign-up. By clicking on the text, users will be directed to the login and sign-up page.
In the 'Body' area, the container is divided into two parts. The left part is the slogan to attract users' attention and demonstrate WorkFlix's main function. Under the slogan, there is a button for users to sign up. The left part is an image banner, which uses an image to highlight the website's using scenario.

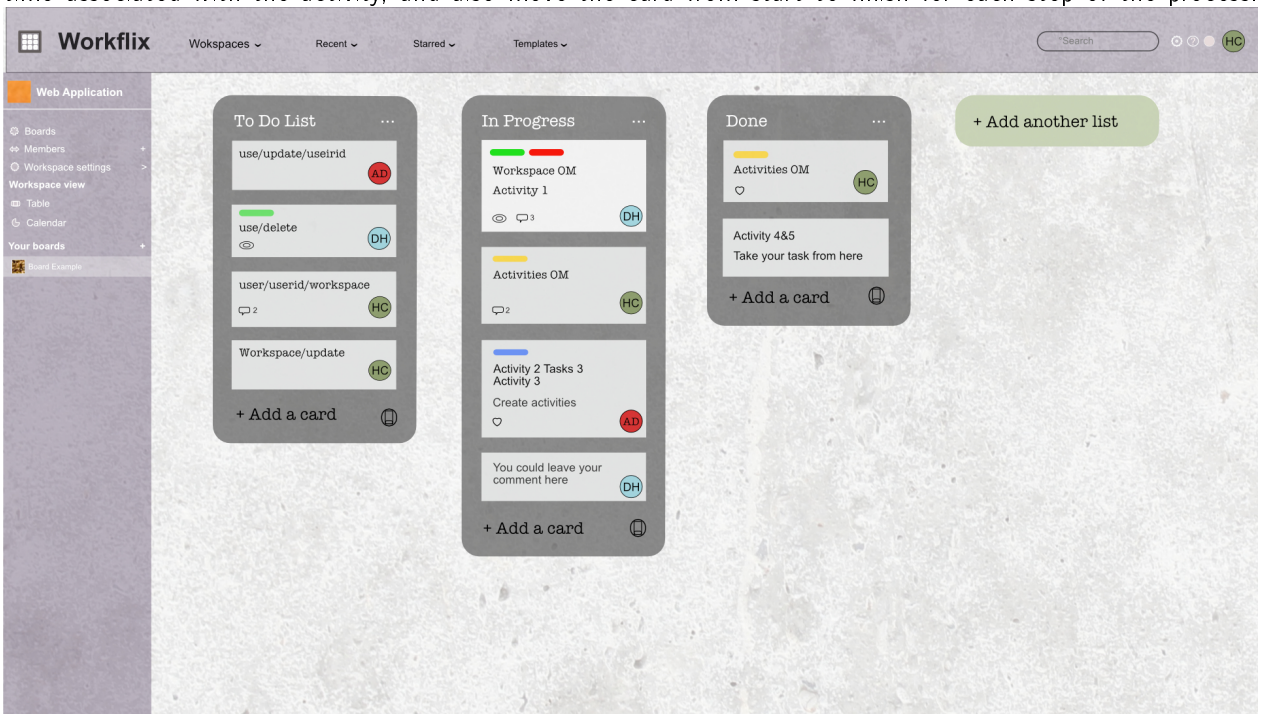
Board Page

A board page is a page created under a workspace to organize tasks. On a board page, users can keep track of information about projects and workflows, which helps you collaborate with your colleagues.

Once users have selected a workspace, they can create different board pages in the workspace A workspace can have multiple boards at the same time. Click on one of them to go to the board page.

On board page users can create lists in their boards. Lists take cards, or specific tasks, and organize them by their various stages of progress. Lists can be used to create a workflow. By clicking "Add another list", entering the list title and clicking Add List, you can add your new list to your board page.

Users can create the required tasks to the list by adding cards.Just click "Add a card..." at the bottom of any list to create a new card, and give it a name.Cards can be customized to hold a wide variety of useful information by clicking on them. By clicking the Edit button, you can add, delete and change the activity, for example, modify the activity label, add or delete members associated with the activity, add or delete time associated with the activity, and also move the card from start to finish for each step of the process.



Data Logic Layer

Entity-Relationship Schema

The entity-relationship contains 8 main entities:

- Maintenance_event: describes maintenance events recorded by the fictitious ride constructor. The primary key is an integer with auto-increment. There are two additional attributes: the description, which is a CHAR field of variable length and the type, which is of type event categories (custom enumeration). To accountability reasons, the email of the user inserting the maintenance is also recorded. This corresponds to the relation "performed" in the ER schema. By being a 1-N relation, this information is recorded directly in the maintenance_event tuple.
- User: each user has, as primary key, their email, which is of type char with variable length (up to 64 characters). For each user we also record their first name, last name (of type char), role (of type roles) and password. Note that the password is hashed through md5 before storing it.
- Park: each park is uniquely identified by its name and has 2 attributes: its address and its email (used to contact the staff of the park).
- Model: The models of rides developed by the fictitious corporation. Each model is uniquely identified by a name and has a textual description.
- Ride: the main entity of the database. Its primary key is an integer incremented automatically upon insertion. It has an additional attribute which is its description. Each ride is deployed in a specific park, (determining a relation N-1 with the parks) and is of a specific model (determining a relation N-1 with the models). The park in which the ride is deployed and its model are therefore memorized as external keys directly on the ride entity instances.
- Device: each ride has multiple devices, which are used to collect numerical data on the functioning of the ride itself. Each device is identified uniquely based on an auto increment integer. Each device is characterized by a name and a description. Finally, since they are in relation N-1 with the rides, they contain directly the id of the ride they are mounted on. note that the type of the device of type device types, which is a custom enumeration data type.
- Session: measurements are organized in sessions. Each session contains several measurements, for example one (or more, according to the number of devices mounted on the ride) for each cycle of the ride. Each session is identified uniquely through an integer with auto increment. Other attributes are the start and end time, represented through timestamps, the date it was added, the type of session and a free text field which contains additional descriptive notes. Each session is linked to a specific ride and thus it contains as a foreign key the id of the ride.
- Measurement: this table has two external keys, in particular, it refers to a specific session and a specific device. Each tuple contains information on the measurements collected by a specific device, during a specific session of measurements. There is an attribute, data_array, which contains the array of the measurements collected by the device. There are three more attributes, start_time and end_time which contain respectively the timestamp of the starting moment and ending moment of the measurements, and the attribute cycle, which indicates, in the session, the progressive number of cycles of the measurement.

Business Logic Layer

Class Diagram

The class diagram contains (some of) the classes used to handle three types of resources: users, rides and parks. It is possible to observe that we have a single servlet to handle the parks. Such servlet implements the doGet and doPost methods and is a subclass of the HttpServlet class. Concerning the rides, the servlet that implements the required behaviours to handle such resources is a more traditional servlet, based on the REST paradigm. In particular, we have a servlet, which also handles the Devices resource, which parses the URI, determining the type (and possibly the id) of the resource that the user wants to interact with. Once the servlet has processed the request, it forwards it to the proper Rest manager class. There are two rest manager classes: RideRestResource and DeviceRestResource. Both are sub-classes of RestResource and implement the methods to handle the proper resource. Concerning the user, there are multiple servlets used to handle the user resource. In particular, we have a servlet (UserServlet) which is mapped to URIs which are freely accessible from outside. This servlet is used to login and register new users. It has a subclass, UserServlet, which overrides the method "RegistrationOperations", by also sending an email to the user upon registration. Note that UserServlet has been kept mainly for legacy reasons. Additionally, the UserServlet implements methods which are accessible only to users with the "admin" authorization level. In particular, it is responsible for the retrieval, update and deletion of users. Several Data Access Objects (DAOs) allow us to obtain different resources. We report the Classes that allow to handle objects of different types.

REST API Summary

The rest API is studied to experiment with multiple different approaches to the REST paradigm. More in detail, endpoints associated with rides and devices are developed in a more traditional way, with the entirety of the information needed to satisfy the request directly in the URI. Other endpoints require additional information (such as the type of operation required) as additional parameters in the request.

Part of the URIs are filtered through different filters. M indicates that only users with the role of Manager can access to the endpoint, E indicates that only editors can access the endpoint, A that only administrators can request the specified URI to the server.

URI	Method	Description	Filter
user/login/	POST	Enables passing the web server the credential of a user. If the credentials are correct it provides a JWT token.	U
user/register/	POST	Enables a user to register for the app.	U
user/logout/	POST	Logout a user	U
user/delete/userid	DELETE	Enables an authenticated user to delete his account. Login information should be given again before proceeding to the deletion.	U
user/update/userid	GET	Enable an authenticated user to update his personal information.	U

user/update/userid/password	POST	Enable an authenticated user to update his password	U
user/userid	POST	Get user information	U
template/get	GET	Get templates to use for a workspace	U
template/create	POST	Enable an authenticated user to create a new template.	U
template/delete/templateid	DELETE	Enable an authenticated user to delete one of his templates.	U
template/update/templateid	PUT	Enable an authenticated user to update one of his templates.	U
user/userid/workspaces	GET	Returns all the workspace in which the authenticated user is connected.	U
workspace/workspaceid	GET	Returns the specific information about a single workspace	U
workspace/create/workspaceid	POST	Enables an authenticated user to create a new workspace. When a user creates a workspace he is the manager of that workspace.	U
workspace/delete/workspaceid	DELETE	Allows the manager of a workspace to delete it.	M
workspace/update/workspaceid	PUT	Allows the manager of a workspace to update it	M
workspace/adduser	POST	Enable the manager of a workspace to add someone to a workspace	M
workspace/removeuser	DELETE	Enable the manager of a workspace to remove someone from the workspace	M
workspace/assignuserpermission	POST	Enable the manager of a workspace to assign different kinds of permissions to a user in a workspace	U
board/boardid	GET	Returns all the information related to a board	M
workspace/workspaceid/boards	GET	Returns all the boards related to a workspace	M
board/create	POST	Allows manager and editor of a workspace to create a board	E
board/delete/boardid	DELETE	Allows to pass to the ws of a usnd their mail and password correspond.	E
board/update/boardid	PUT	Allows manager and editor of a workspace to edit a board by {board id}	E
subboard/subboardid	GET	Allows manager and editor of a workspace to delete a board by {board id}	U

board/boardid/subboards	GET	Returns all subboard data belonging to the current board	U
subboard/create	POST	Allows to create a new subboard	E
subboard/delete/subboardid	DELETE	Allows to delete the subboard by its {subboard id}	E
subboard/update/subboardid	PUT	Allows to update the subboard by its {subboard id}	U
subboard/subboardid/activities	GET	Returns all activitydata belonging to the current subboard with the {subboard id}	U
activity/activityid	GET	Returns all subboard data by its {activity id}	U
activity/create	POST	Allows to create a new activity	U
activity/delete/activityid	DELETE	Allows to delete the activityby its {activity id}	U
activity/update/activityid	PUT	Allows to update the activityby its {activity id}	U
activity/assignee/get	GET	Returns the assignee data of the current activity	U
activity/assignee/add	PUT	Allows to insert new assignee of the current activity	U
activity/assignee/remove	POST	Allows to delete assignee of the current activity	U
activity/comment/get	GET	Returns comment	U
activity/comment/create	PUT	Allows to create new comment	U
activity/comment/delete/commentid	DELETE	Allows to insert comment by its {comment id}	U
activity/comment/update/commentid	PUT	Allows to update comment by its {comment id}	U
analytics/get	GET	Returns analytics data.	M

Table 2: Main REST API entrypoints

REST Error Codes

Here the list of errors defined in the application. Application specific errors have the application error which follows a progressive numeration starting from -100. METHOD NOT ALLOWED errors are identified with the error code -500. Internal errors, which correspond to crashes, servlet exceptions, or problems with the input/output streams are identified with the Error Code -999.

Error Code	HTTP Status Code	Description
-200	OK	Indicates that the request has succeeded.

-201	Created	Indicates that the request has succeeded and a new user/workspace/board/subboard/comment has been created as a result.
-400	BAD_REQUEST	Request made to the server without specifying all the mandatory params
-401	Unauthorized	Failed login attempt due to unauthorized user information.
-403	Forbidden	Unauthorized request. The client does not have access rights to the content.
-404	NOT_FOUND	The server can not find the requested resource. Use a broken link or a mistyped URL.
-409	CONFLICT	mail already used.
-409	CONFLICT	The username has already been used.
-409	CONFLICT	The template name has already been used.
-500	Internal_Server_Error	The server encountered an unexpected condition that prevented it from fulfilling the request.

Table 3: Error codes for the REST interface of the Amusement Park application back-end

Group Members Contribution

- Huimin Chen: landing page design, document composition,template and analytics API
-
-