UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

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 WF	Project WorkFlix Web App	
First Name	Last Name	Student Number
Huimin	Chen	2071518
Alessandro	Discalzi	2088235
Reza	Khaleghi	2080242
Pedram	Zeinalabedin Zadegan	2072548
Amirhossein	Kargar Khabbazi Sardroud	2072851
Esra	Erdim	2073512
Huan	chen	2071513

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.

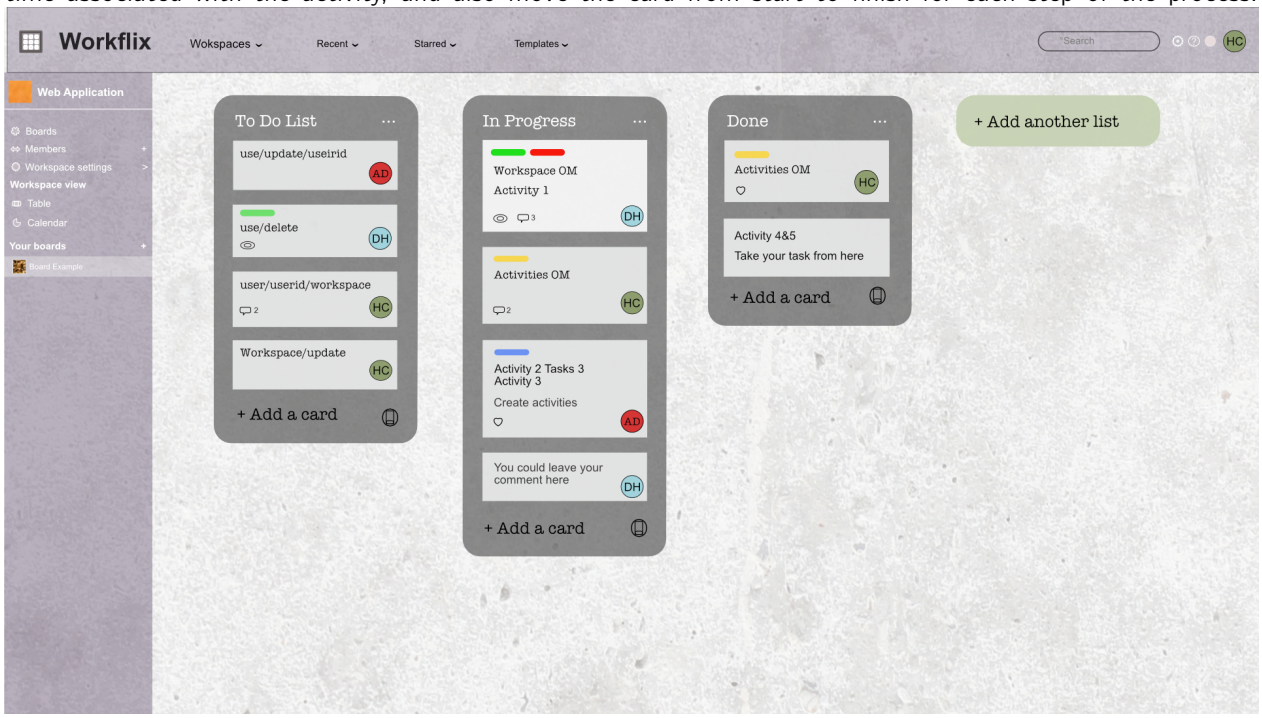
Board Page

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.



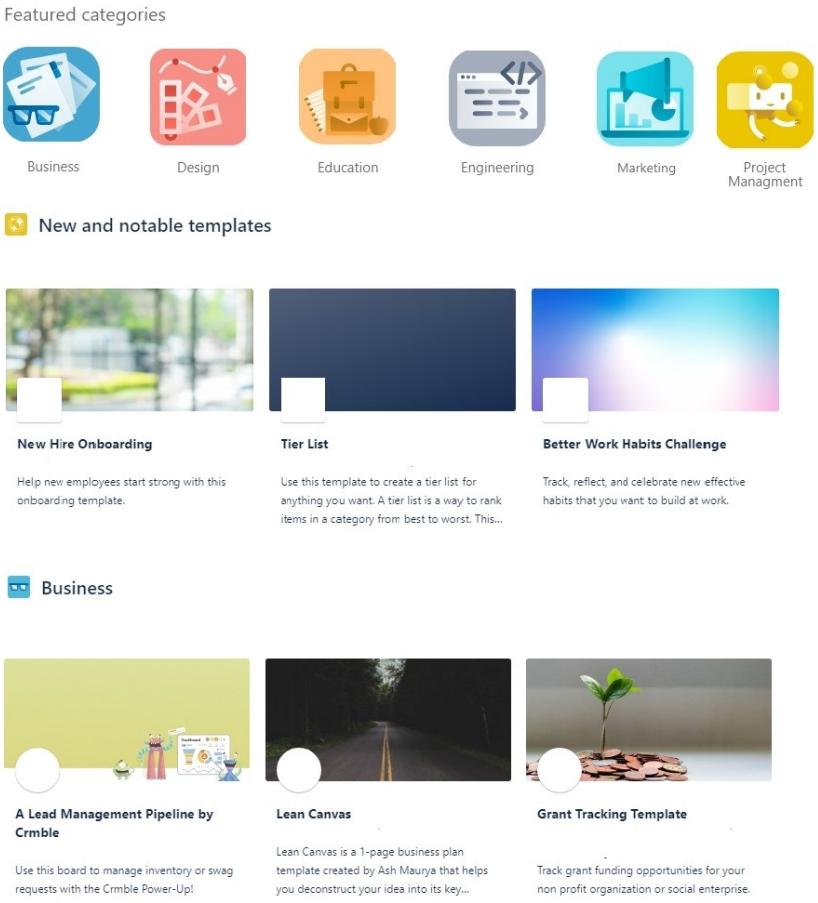
Template Page

A template in a web application refers to a pre-designed layout or structure that can be customized to create a new web page or website.

Templates typically include the basic elements of a web page, such as headers, footers, navigation menus, and content sections, which can be modified to fit the specific needs of the user.

Templates can be created from scratch, or they can be downloaded from a library of pre-built templates, which are often available for free or for a fee.

Web developers and designers use templates to save time and streamline the web development process.



User Setting

User settings in a web application refer to the customizable options and preferences that users can set to personalize their experience within the application.

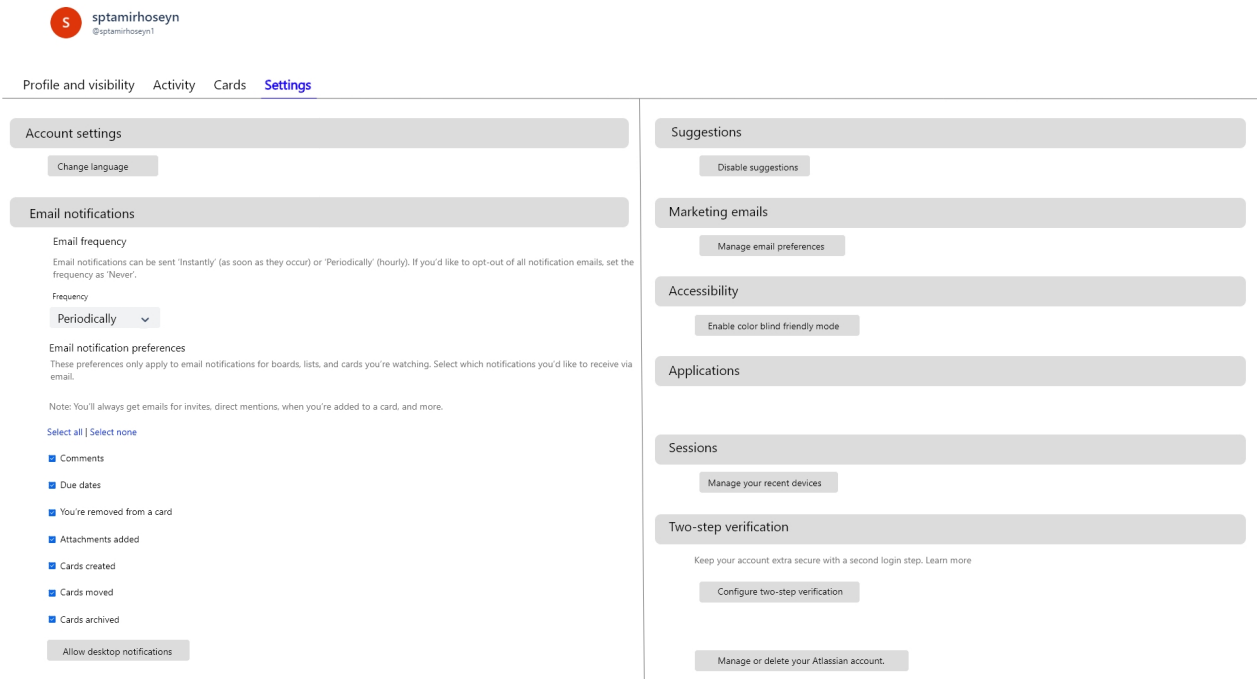
User settings may include options such as language preferences, time zone settings, font size, color schemes, notification preferences, privacy and security settings, and more.

These settings can enhance the user experience by allowing users to tailor the application to their specific needs and preferences.

User settings are typically accessed via a user profile or settings page within the web application and can be modified at any time by the user.

User settings can also include features like account management, password and email settings, and social media integrations.

By providing users with control over their experience, web applications can improve user engagement and satisfaction.



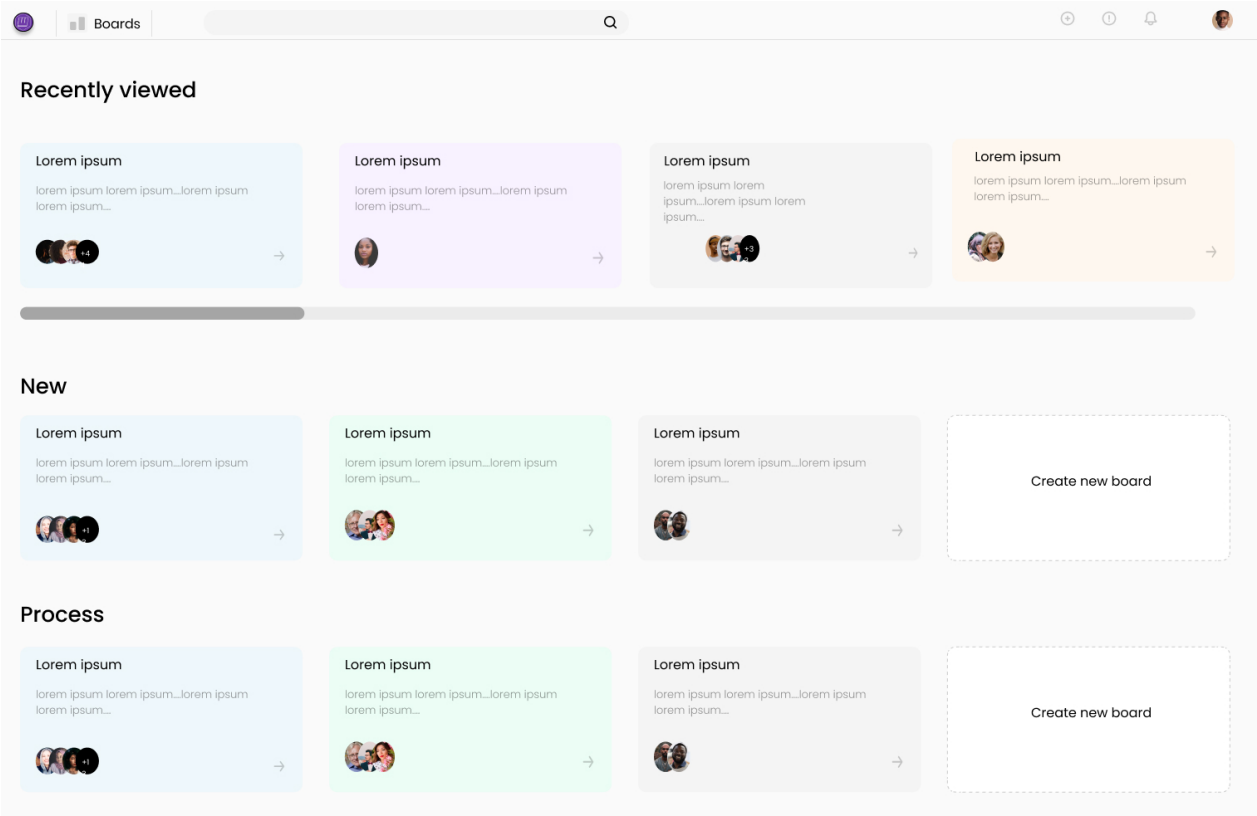
Workspace

In a web application, a workspace refers to a virtual environment where users can collaborate and work on projects, tasks, or documents together.

Workspaces typically include tools for communication, file sharing, project management, and other collaborative features that allow users to work together in real-time.

Workspaces can be used for a variety of purposes, such as team collaboration, project management, online learning, and more. Users can typically create their own workspaces or be invited to join existing ones by other users or administrators.

Workspaces can be accessed via a web browser or mobile application and can be customized to fit the specific needs of the users.



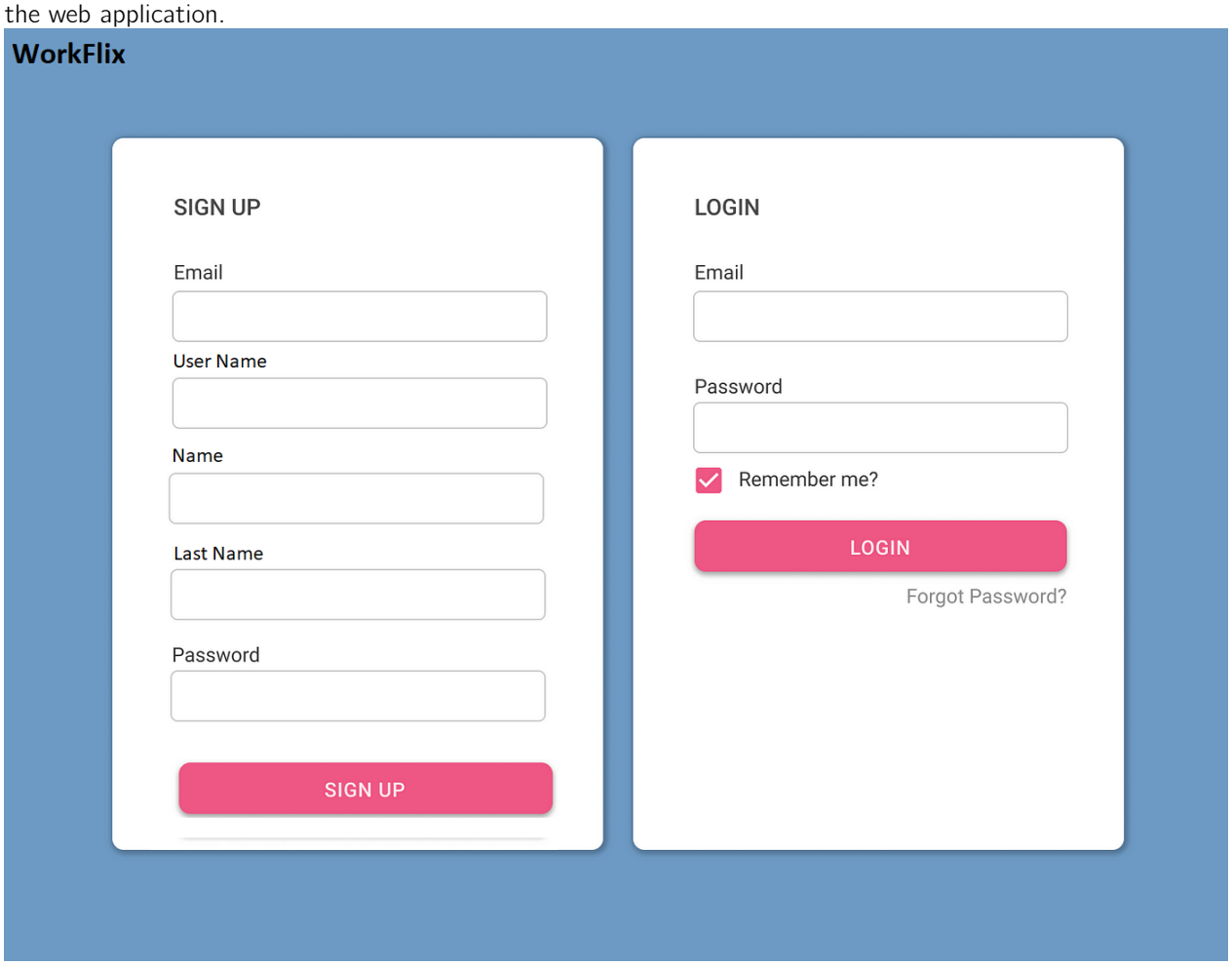
Sign iN Login

In a web application, "sign in" or "login" refers to the process of accessing a user's account within the application by providing the appropriate credentials, such as a username or email address and password.

The purpose of the sign-in process is to authenticate the user and grant access to the application's features and content that are specific to the user's account.

Typically, the sign-in or login page is the first page a user encounters when accessing a web application and requires the user to enter their credentials before proceeding to the main dashboard or homepage.

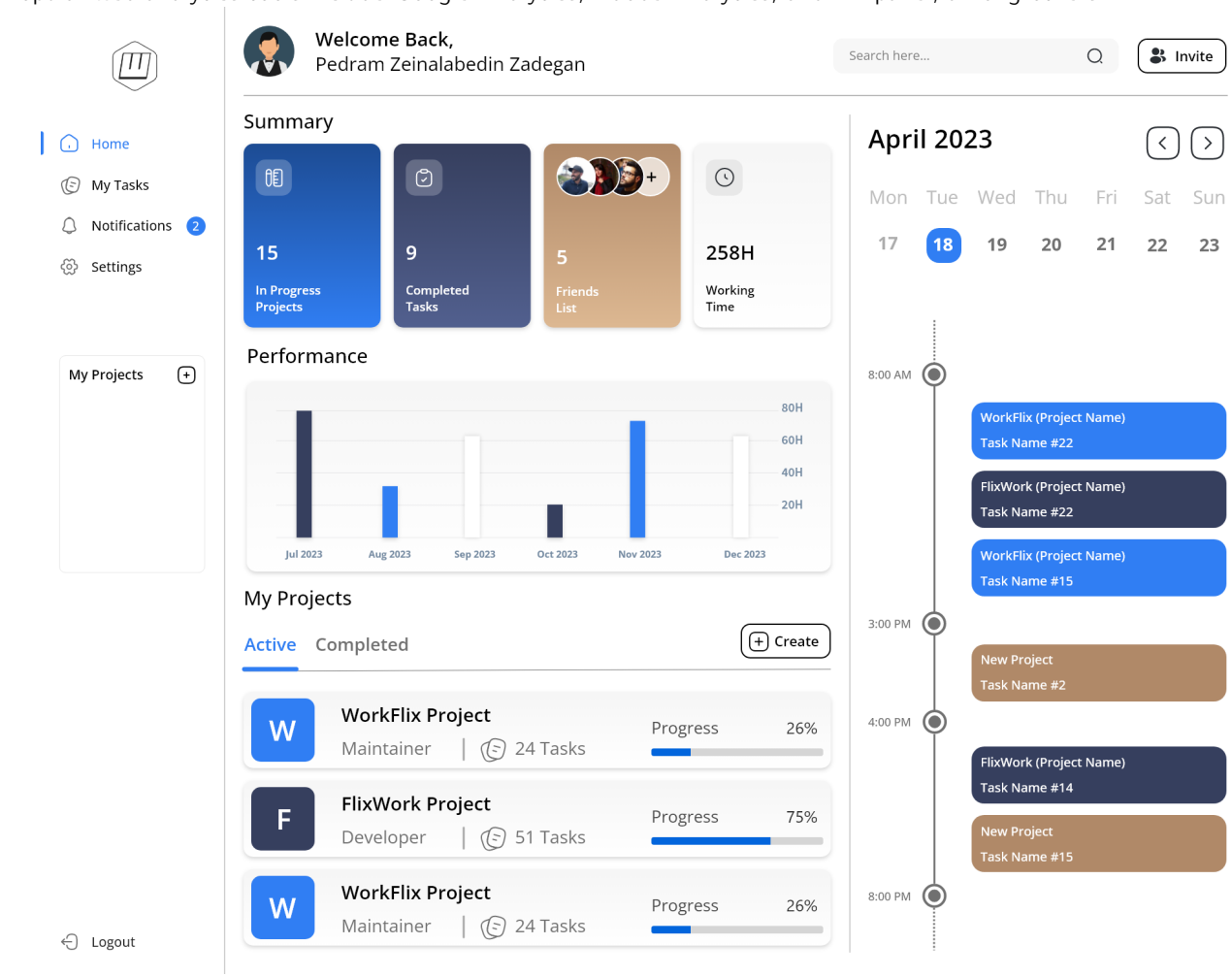
The sign-in process is essential for ensuring the security and privacy of the user's account and data within the web application.



Analytics

analytics refers to the collection, measurement, and analysis of data related to user behavior and activity within the application.
Analytics tools are used to track user interactions with the application, such as page views, clicks, downloads,

and other actions that can help developers and business owners understand how users are engaging with the application.
Analytics data can also provide insights into user demographics, preferences, and behaviors, which can be used to improve the user experience, optimize marketing campaigns, and make data-driven decisions about the application's development and performance.
Popular web analytics tools include Google Analytics, Adobe Analytics, and Mixpanel, among others.

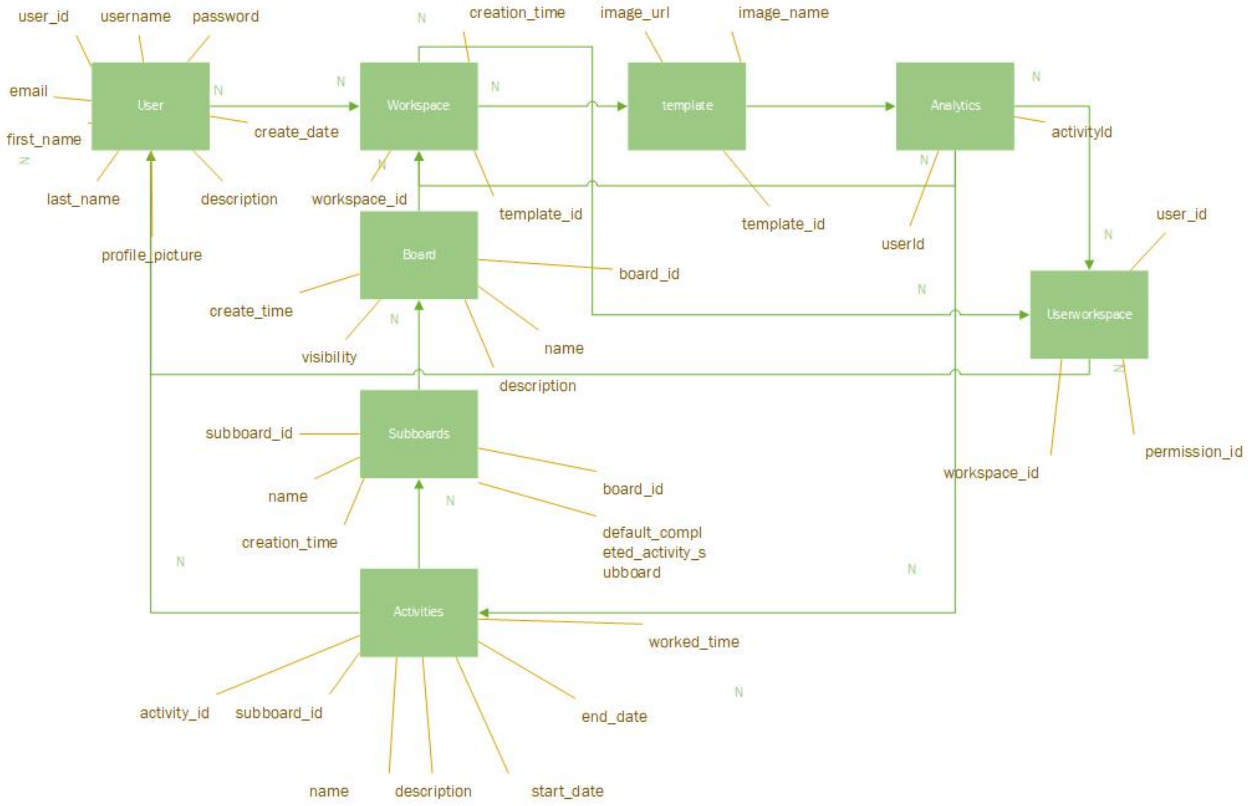


Data Logic Layer

Entity-Relationship Schema

The entity-relationship contains 8 main entities:

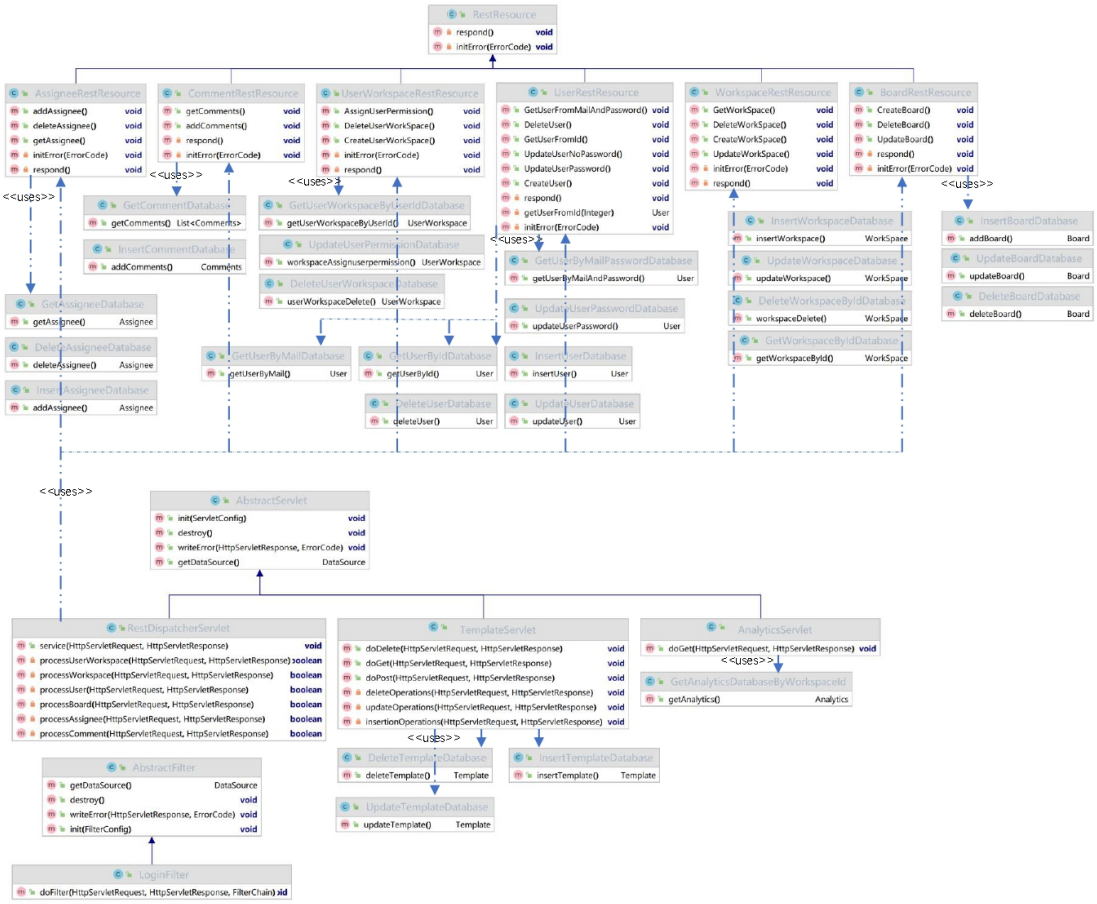
- User: each user has, as primary key, their email, which is of type char with variable length (up to 64 characters). For each user we also record their first name, last name (of type char), role (of type roles) and password. Note that the password is hashed through md5 before storing it.
- Workspace: a workspace is a virtual space where teams can collaborate on projects and tasks. It is essentially a container for multiple boards, which are individual task lists that can be customized to fit the needs of specific projects or workflows. Each model is uniquely identified by a workspace-id and has a textual description and permission-name.
- User Workspace: the user work space is the place for controlling the users,identified by its user-id, workspace-id, permission-id.
- Board: a board is the main organizational unit used to manage tasks and projects and identified by board-id and has create-time, description, visibility, name.
- Analytics: each model of Analytics, analyze the performances of each user in working and provide report. It is identified by user-id and activity-id.
- Subboard: each model is a board which is sub-part of board and identified by subboard-id and has board-id and name
- Activities:activities refer to the actions or events that have occurred on a board or card.it's identified by activity-id and it has subboard-id and name and description
- Template:a template refers to a pre-designed board setup that can be used as a starting point for creating new boards. this table has two external keys, it identified as template-id and has imag-url and image-name


user_id
username
password
email
first_name
last_name
description
profile_picture
User
create_date
Workspace
creation_time
workspace_id
template_id
image_url
image_name
template
template_id
Analytics
activityId
userId
Board
board_id
create_time
visibility
name
description
Userworkspace
user_id
workspace_id
permission_id
Subboards
subboard_id
name
creation_time
board_id
default_compl
eted_activity_s
ubboard
Activities
worked_time
activity_id
subboard_id
name
description
start_date
end_date
N

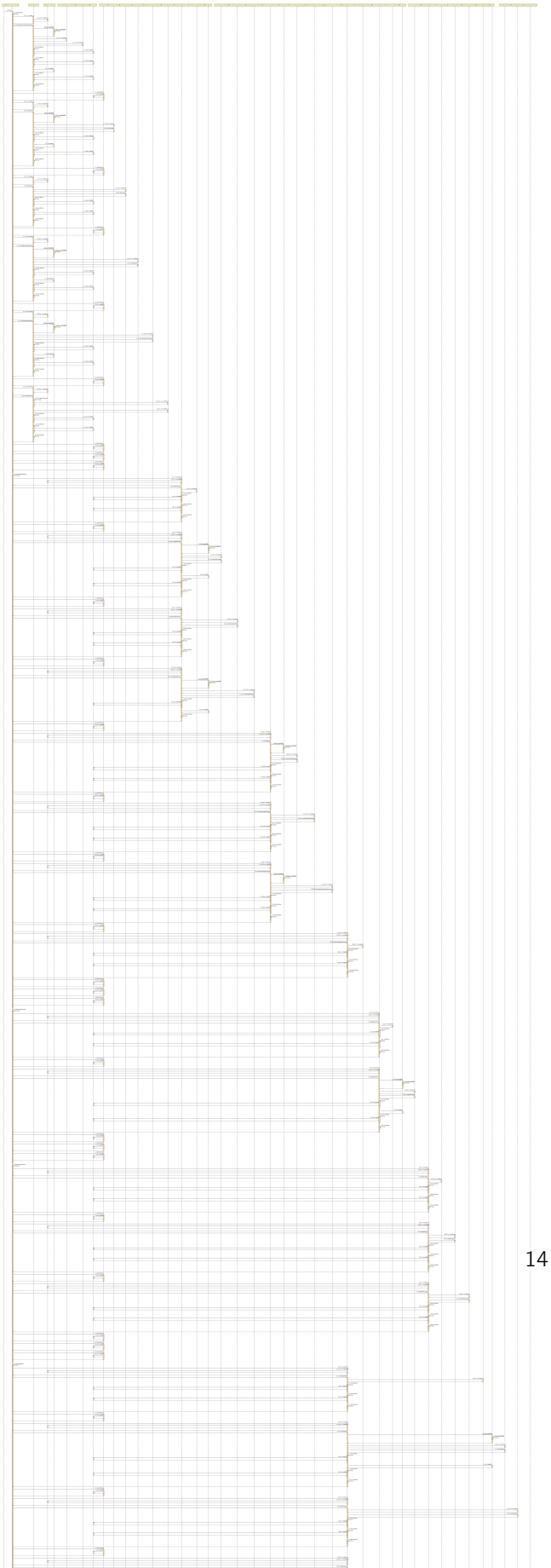
Business Logic Layer

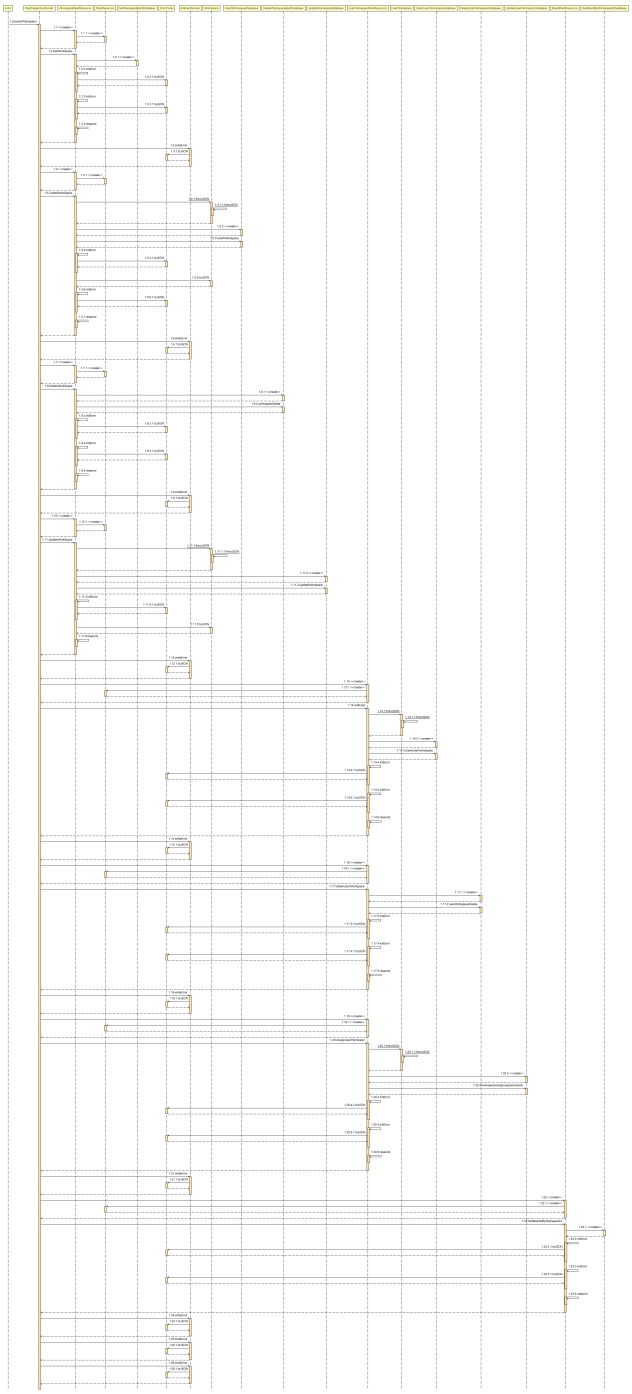
Class Diagram

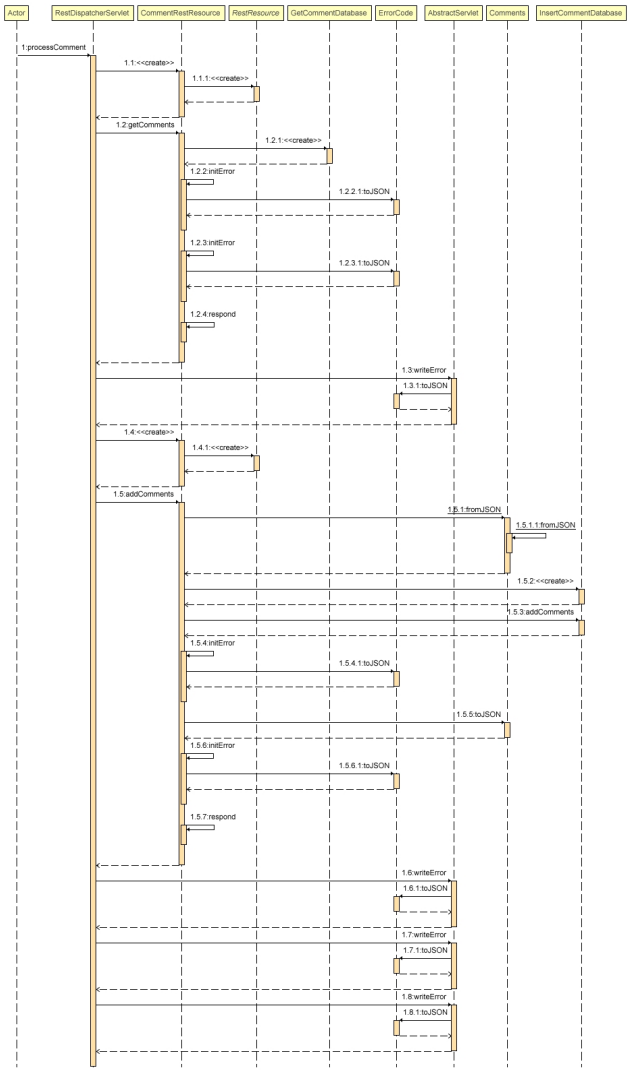
The class diagram contains (some of) the classes used to handle three types of resources: users, rides and parks. It is possible to observe that we have a single servlet to handle the parks. Such servlet implements the doGet and doPost methods and is a subclass of the HttpServlet class. Concerning the 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.

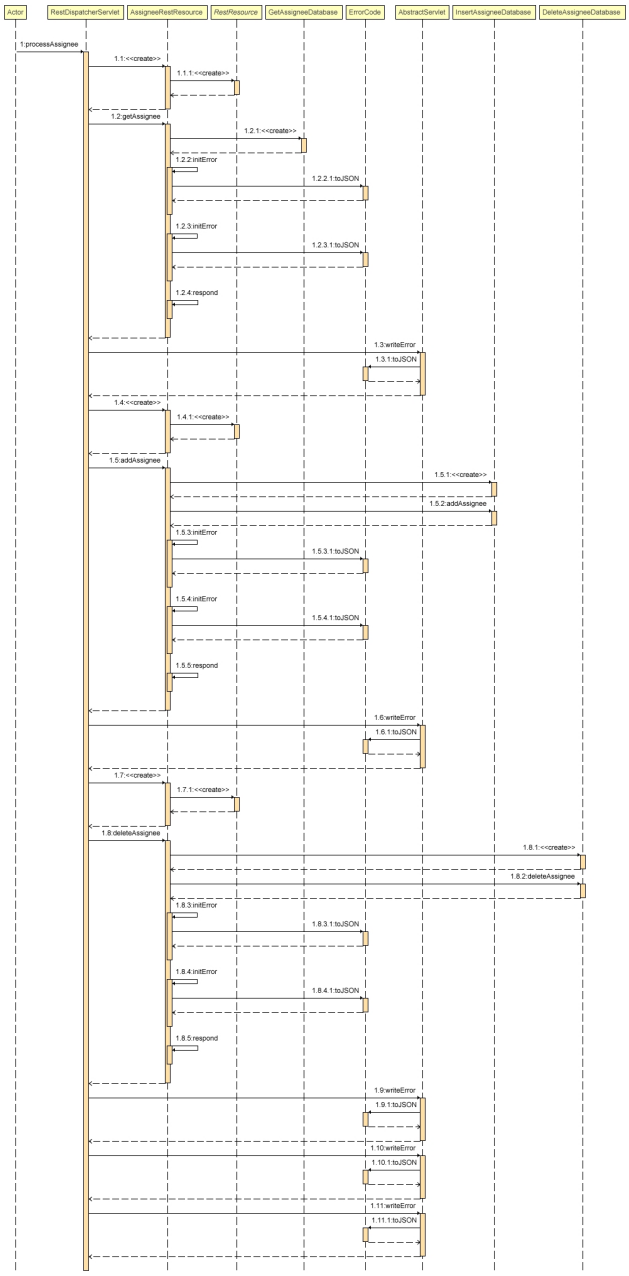

RestResource
respond() void
initError(ErrorCode) void
AssigneeRestResource
addAssignee() void
deleteAssignee() void
getAssignee() void
initError(ErrorCode) void
respond() void
CommentRestResource
getComments() void
addComments() void
respond() void
initError(ErrorCode) void
UserWorkspaceRestResource
AssignUserPermission() void
DeleteUserWorkSpace() void
CreateUserWorkSpace() void
initError(ErrorCode) void
respond() void
UserRestResource
GetUserFromMailAndPassword() void
DeleteUser() void
GetUserFromId() void
UpdateUserNoPassword() void
UpdateUserPassword() void
CreateUser() void
respond() void
getUserFromId(Integer) User
initError(ErrorCode) void
WorkspaceRestResource
GetWorkSpace() void
DeleteWorkSpace() void
CreateWorkSpace() void
UpdateWorkSpace() void
initError(ErrorCode) void
respond() void
BoardRestResource
CreateBoard() void
DeleteBoard() void
UpdateBoard() void
respond() void
initError(ErrorCode) void
<<uses>>
GetCommentDatabase
getComments() List<Comments>
InsertCommentDatabase
addComments() Comments
GetUserWorkspaceByUserIdDatabase
getUserWorkspaceByUserId() UserWorkspace
UpdateUserPermissionDatabase
workspaceAssignuserpermission() UserWorkspace
DeleteUserWorkspaceDatabase
userWorkspaceDelete() UserWorkspace
GetUserByMailPasswordDatabase
getUserByMailAndPassword() User
InsertWorkspaceDatabase
insertWorkspace() WorkSpace
UpdateWorkspaceDatabase
updateWorkspace() WorkSpace
DeleteWorkspaceByIdDatabase
workspaceDelete() WorkSpace
GetWorkspaceByIdDatabase
getWorkspaceById() WorkSpace
InsertBoardDatabase
addBoard() Board
UpdateBoardDatabase
updateBoard() Board
DeleteBoardDatabase
deleteBoard() Board
GetAssigneeDatabase
getAssignee() Assignee
DeleteAssigneeDatabase
deleteAssignee() Assignee
InsertAssigneeDatabase
addAssignee() Assignee
GetUserByMailDatabase
getUserByMail() User
GetUserByIdDatabase
getUserById() User
InsertUserDatabase
insertUser() User
DeleteUserDatabase
deleteUser() User
UpdateUserPasswordDatabase
updateUserPassword() User
UpdateUserDatabase
updateUser() User
AbstractServlet
init(ServletConfig) void
destroy() void
writeError(HttpServletResponse, ErrorCode) void
getDataSource() DataSource
RestDispatcherServlet
service(HttpServletRequest, HttpServletResponse) void
processUserWorkspace(HttpServletRequest, HttpServletResponse) boolean
processWorkspace(HttpServletRequest, HttpServletResponse) boolean
processUser(HttpServletRequest, HttpServletResponse) boolean
processBoard(HttpServletRequest, HttpServletResponse) boolean
processAssignee(HttpServletRequest, HttpServletResponse) boolean
processComment(HttpServletRequest, HttpServletResponse) boolean
TemplateServlet
doDelete(HttpServletRequest, HttpServletResponse) void
doGet(HttpServletRequest, HttpServletResponse) void
doPost(HttpServletRequest, HttpServletResponse) void
deleteOperations(HttpServletRequest, HttpServletResponse) void
updateOperations(HttpServletRequest, HttpServletResponse) void
insertionOperations(HttpServletRequest, HttpServletResponse) void
AnalyticsServlet
doGet(HttpServletRequest, HttpServletResponse) void
GetAnalyticsDatabaseByWorkspaceId
getAnalytics() Analytics
DeleteTemplateDatabase
deleteTemplate() Template
InsertTemplateDatabase
insertTemplate() Template
UpdateTemplateDatabase
updateTemplate() Template
AbstractFilter
getDataSource() DataSource
destroy() void
writeError(HttpServletResponse, ErrorCode) void
init(FilterConfig) void
LoginFilter
doFilter(HttpServletRequest, HttpServletResponse, FilterChain) void

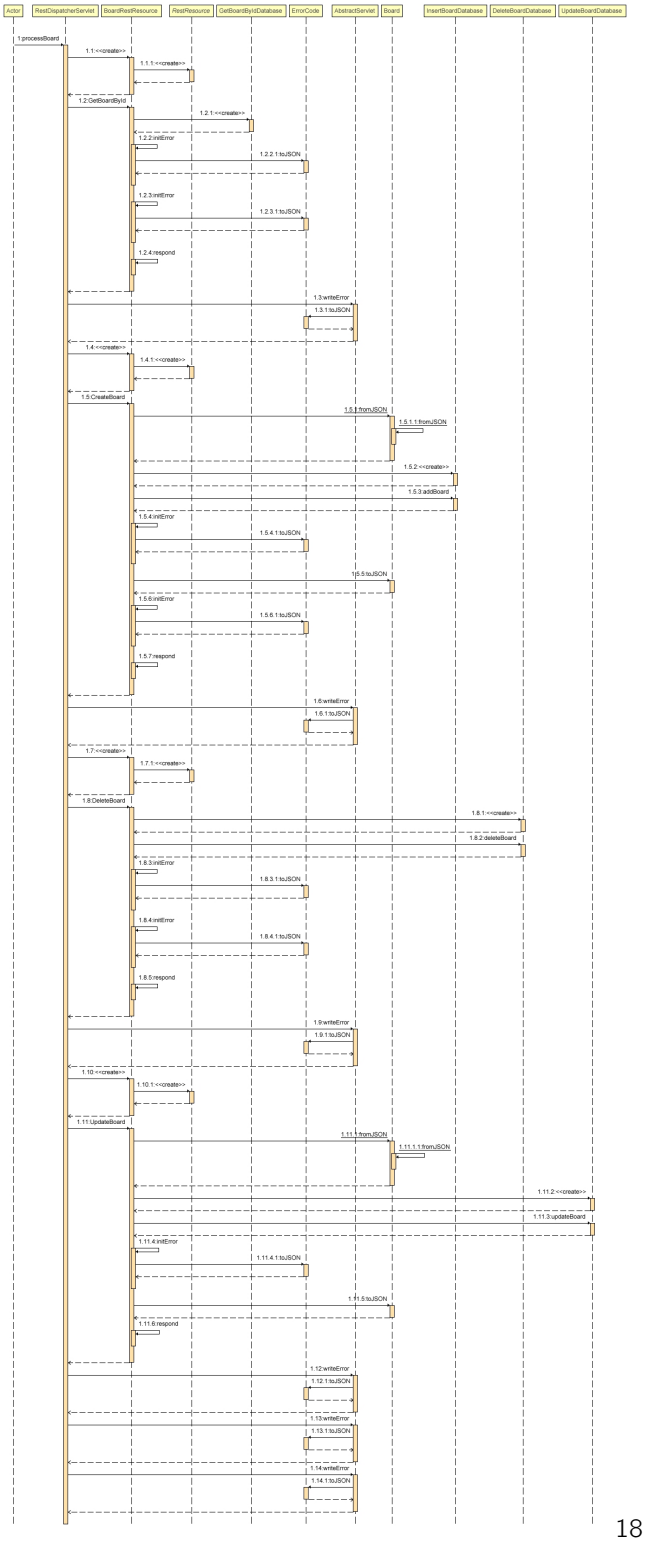
Sequence Diagrams






Actor
RestDispatcherServlet
CommentRestResource
RestResource
GetCommentDatabase
ErrorCode
AbstractServlet
Comments
InsertCommentDatabase
1:processComment
1.1:<<create>>
1.1.1:<<create>>
1.2:getComments
1.2.1:<<create>>
1.2.2:initError
1.2.2.1:toJSON
1.2.3:initError
1.2.3.1:toJSON
1.2.4:respond
1.3:writeError
1.3.1:toJSON
1.4:<<create>>
1.4.1:<<create>>
1.5:addComments
1.5.1:fromJSON
1.5.1.1:fromJSON
1.5.2:<<create>>
1.5.3:addComments
1.5.4:initError
1.5.4.1:toJSON
1.5.5:toJSON
1.5.6:initError
1.5.6.1:toJSON
1.5.7:respond
1.6:writeError
1.6.1:toJSON
1.7:writeError
1.7.1:toJSON
1.8:writeError
1.8.1:toJSON


Actor
RestDispatcherServlet
AssigneeRestResource
RestResource
GetAssigneeDatabase
ErrorCode
AbstractServlet
InsertAssigneeDatabase
DeleteAssigneeDatabase
1:processAssignee
1.1:<<create>>
1.1.1:<<create>>
1.2:getAssignee
1.2.1:<<create>>
1.2.2:initError
1.2.2.1:toJSON
1.2.3:initError
1.2.3.1:toJSON
1.2.4:respond
1.3:writeError
1.3.1:toJSON
1.4:<<create>>
1.4.1:<<create>>
1.5:addAssignee
1.5.1:<<create>>
1.5.2:addAssignee
1.5.3:initError
1.5.3.1:toJSON
1.5.4:initError
1.5.4.1:toJSON
1.5.5:respond
1.6:writeError
1.6.1:toJSON
1.7:<<create>>
1.7.1:<<create>>
1.8:deleteAssignee
1.8.1:<<create>>
1.8.2:deleteAssignee
1.8.3:initError
1.8.3.1:toJSON
1.8.4:initError
1.8.4.1:toJSON
1.8.5:respond
1.9:writeError
1.9.1:toJSON
1.10:writeError
1.10.1:toJSON
1.11:writeError
1.11.1:toJSON


Actor
RestDispatcherServlet
BoardRestResource
RestResource
GetBoardById1Database
ErrorCode
AbstractServlet
Board
InsertBoardDatabase
DeleteBoardDatabase
UpdateBoardDatabase
1:processBoard
1.1:<<create>>
1.1.1:<<create>>
1.2:GetBoardById
1.2.1:<<create>>
1.2.2:initError
1.2.2.1:toJSON
1.2.3:initError
1.2.3.1:toJSON
1.2.4:respond
1.3:writeError
1.3.1:toJSON
1.4:<<create>>
1.4.1:<<create>>
1.5:CreateBoard
1.5.1:fromJSON
1.5.1.1:fromJSON
1.5.2:<<create>>
1.5.3:addBoard
1.5.4:initError
1.5.4.1:toJSON
1.5.5:toJSON
1.5.6:initError
1.5.6.1:toJSON
1.5.7:respond
1.6:writeError
1.6.1:toJSON
1.7:<<create>>
1.7.1:<<create>>
1.8:DeleteBoard
1.8.1:<<create>>
1.8.2:deleteBoard
1.8.3:initError
1.8.3.1:toJSON
1.8.4:initError
1.8.4.1:toJSON
1.8.5:respond
1.9:writeError
1.9.1:toJSON
1.10:<<create>>
1.10.1:<<create>>
1.11:UpdateBoard
1.11.1:fromJSON
1.11.1.1:fromJSON
1.11.2:<<create>>
1.11.3:updateBoard
1.11.4:initError
1.11.4.1:toJSON
1.11.5:toJSON
1.11.6:respond
1.12:writeError
1.12.1:toJSON
1.13:writeError
1.13.1:toJSON
1.14:writeError
1.14.1:toJSON

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

REST API DETAILS

In WorkFlix web application project, we have implemented several apis using servlet and REST. In this section, two apis, as examples, are explained with details.

Users

The following endpoint allows to register a new user.

- URL: `user/register`
- Method: `POST`
- URL Parameters: No parameters are required in the url
- Data Parameters:

  Required:

  - `username = {string}`
  - `password = {string}`
  - `email = {string}`
  - `first_name = {string}`
  - `last_name = {string}`
  - `profile_picture={string}`
  - `description {string}`
  - `create_date ={string}`

- Success Response:

  Code: 200
  Content: the user is redirected to the login page, which in turns uses a filter to create a session for the user and redirect them to the homepage.

- Error Response:

  Code: 409 CONFLICT
  Content: `{‘‘error’’: {‘‘code’’: -101, ‘‘message’’ : ‘‘User already exists.’’}`
  When: the user is trying to register with a user already present in the database.

  Code: 500 INTERNAL_SERVER_ERROR
  Content: `{‘‘error’’: {‘‘code’’: -999, ‘‘message’’ : ‘‘Internal Error.’’}`
  When: if there is a `SQLException` or a `NamingException`, this error is return.

Template
The following endpoint allows to create a template.

- URL: `template/create`
- Method: `POST`
- URL Parameters: none
- Data Parameters:

  Required:

  - `image_url = {string}`
  - `template_name = {string}`

- Success Response:

  Code: 201 CREATED
  Content: `{''error'': {''code'': 0, ''message'' : ''Template inserted correctly''}`

- Error Response:

  Code: 400 BAD_REQUEST
  Content: `{''error'': {''code'': -101, ''message'' : ''Template information missing.''}`
  When: the user is trying to add a new template but the request parameter is missing.

  Code: 409 CONFLICT
  Content: `{''error'': {''code'': -101, ''message'' : ''Template name already exists.''}`
  When: the template name is alsready used in the database.

  Code: 500 INTERNAL_SERVER_ERROR
  Content: `{''error'': {''code'': -999, ''message'' : ''Internal Error.''}`
  When: if there is a `SQLException` or a `NamingException`, this error is return.

Group Members Contribution

- Huimin Chen: landing page design, document composition,template and analytics API
- Alessandro Discalzi: API definition, user APIs and general fixes, document composition,
- Amirhossein Kargar Khabbazi Sardroud: User setting and log in page, Workspace and User workspace API and resource, document compositin, Sequence-diagram

- Pedram Zeinalabedin Zadegan: Comment , Assignee and Activity Resourses , Analytics Page,Comment and Assignee Api, Document composition, Squence diagram
- Reza Khaleghi Subboards and Template, Subboards API and resource, Document composition,Sequence Diagram
- Esra Edrim : Workspace Page Design, Board Api and Resource and Document Composition
- Huan Chen:Board Page design,Document Composition ,Activity and Comment api , subboard rest resource,Class Diagram