CAB302 Software Development - Billboard Management System: Report

Author: Hieu Nghia Huynh (N10315071), Jamie Martin (N10212361), Perdana Bailey (N10245065), Trevor Waturuocha (N10209921)

Group Number: 21

Introduction

Information systems are vital to effectively managing inefficient processes whilst remaining scalable. Prior to this adoption, organisations would have to manually manage their data. As a business practice, this can be seen as costly, time-consuming, and unreliable to maintain. An unnamed client corporation approached a team of developers to address a similar problem through the design and development of a system application that can handle electronic billboards. The application was specified to be separated into three distinct components; a billboard viewer, billboard server and billboard control panel. In the back-end, the server was responsible for discretely storing and sorting all billboard related information to be interacted by a user. On the other hand, the viewer displays the billboards on any screen and the control panel facilitates an interactive interface for users to carry out their various administrative tasks. Through the use of industry-standard project management tools such as Atlassian Jira and effective software design, the group was able to quickly deliver a solution that satisfied the client’s requirement.

Terms + Abbreviations

Term	**Description**
Client	An instance of the client package; the Control Panel GUI.
Server	An instance of the server package; the http-like socket-based API.
Viewer	An instance of the viewer package; the Billboard Viewer GUI.
Session, Billboard(s), Schedule(s), User(s) and Permissions	These relate to the class instance.
Models	These refer to the Billboard, Schedule, User and Permissions.
user	The person interacting with the system
Controller	A collection of routes, coupled in a meaningful way, generally by the Model type.
Route	A http-like route, which is tied to a list of Actions.
Action(s)	Singularly: A single method. Pluraly: A list of methods chained together.
TDD	Test-Driven Development.
Request	Refers to the Request model which is received and handled by the Server.
Response	Refers to the Response model which is received and handled by the Client and Viewer.
Bad Request, Internal Server Error, Not Found, Success, Unauthorized, Unsupported Type	Refer to the type of Response

Statement Of Completeness

High-level overview of what features were implemented in your submitted project.

In terms of completeness, the team was able to deliver a solution that enables the company to effectively create, manage and schedule billboards. The next few paragraphs will walk you through a high-level overview of what features were implemented based on the three core components: the Client, the Server and the Viewer.

Client

Upon starting, the Client will gather the required port and address information from the network.props file. This enables the client to initiate connections with the server. These connections are made through an abstracted factory that takes in Request parameters and handles the connection between the server. It returns a Result which can be used wherever necessary.

In terms of functionality, the Client, after startup, spawns a Login frame which prompts the user to login with a username and password. On submission, the password will then be hashed and sent to the server to authenticate. If successful the Client receives a Session which is used to specify what is rendered on the client and as well validate Requests when submitted to the server.

The client holds Billboard, Schedule and User tabs which are rendered based on the users Permissions. When creating Models, dialogs are spawned with the necessary input fields to capture the data of a new entry. Edits can then be made to update selected entries.

An update password and logout button are located on the top right of the Client frame. These allow the user to change their password and end the current users Session.

Server

Upon starting, the Server will read the port required to operate on from the network.props file. The Server will then listen on that port for connections from the Viewer and Client.

The Server allows socket-based connections to submit a Request object which is used to understand the request the user wishes to make, similar to a HTTP request.

All necessary Billboard, Schedule, User and Permissions Controllers have been made to handle the functionality as requested in the specification. These, along with authenticated and permission-blocked middleware, enable the server to act as a secure and http-like REST API.

Viewer

Upon starting, the Viewer will gather the required port and address information from the network.props file. This enables the Viewer to initiate connections with the server.

The Viewer polls the Server every 15 seconds for currently scheduled Billboards. If a Billboard is returned it handles the Billboards variables to render it as specified in the examples given on Blackboard. If there is no Billboard it renders a generic Billboard saying there is no currently scheduled Billboard. The viewer handles connection errors accordingly and displays an error Billboard until a connection is received from the server.

Statement Of Contributions

Names and student numbers of each member of your team, as well as a statement describing what each team member contributed to the final result.

Hieu Nghia Huynh - N10315071

- Developed code for the controller and middleware packages used by the Server.
- Worked on and refined algorithms, such as the scheduling functionality.
- Developed the business model OOP classes and translated those requirements into database SQL scripts.
- Developed unit tests for the controllers and middleware, as well as smaller unit tests where necessary.

Jamie Martin - N10212361

- Developed the group's project management solution through the use of Jira and Bitbucket. Streamlined development efficiency through the use of the Kanban method, enabling the assignment of tasks and better management of the project as a whole.
- Developed ideas and matured structure of lower level components for all areas of the project, such as the sql, components.table and router packages, enabling efficiency through the development of generic, reusable code.
- Refactored code where necessary.
- Worked on various unit tests where required.

Perdana Bailey - N10245065

- Built the team's BitBucket continuous integration pipelines, which ran unit tests in the cloud using a Maven test suite. This enabled automated-testing and alerted the other members when builds failed so that errors were pinched as soon as possible.
- Ensured a standardised and verbose structure, code quality and javadoc across the entire codebase.
- Developed algorithms regarding hashing functionality and login/register functionality.
- Worked on ideas to promote abstraction.

Trevor Waturuocha - N10209921

- Developed the Billboard Viewer, ensuring a strict similarity to what was requested in the specification as well as the examples provided on Blackboard.
- Took part in the higher level decision making for designing the UML diagrams of the application and the GUI elements.
- Developed various GUI views and functionality for the Client.
- Worked on unit tests where required.
- Wrote various sections of the report.

High-Level Design Description

Explaining what each class is responsible for and how it interacts with the other classes in your system.

The group's main focus was developing a clean solution in an effective manner. Focusing on results and building upon these foundations, various sub-packages were developed to deliver and handle core business logic in a segregated but accessible manner for all projects.

Client

components package

The components package holds components built for the Client. This includes a table sub-package which holds all classes required for the generic editable table.

frames package

The frames package holds the high-level logic for the login and main menu of the Client.

panels package

The panels package holds the panels which are rendered on each tab. A majority of the GUI logic and data binding is held in these classes.

services package

The classes in the services package act as singleton controller and state storage between the Client and Server. State is held in these as well as methods the Client can run to send Requests to the Server.

Common

models package

This package holds all the models relating to the business domain.

router package

The router package and classes hold the logic behind the http-like router which enables the mapping of string paths to chainable Actions. These Actions are an extension of the abstract Action class which override an execute method that is executed by the router returning an Ok if successful, otherwise a contextual response that lets the user know where they have gone wrong.

All exceptions that aren't already handled are thrown up to the router.

sql package

The sql package is a simple ORM tool to better manage the connection between the Server and the sqlite database. This ORM tool manages a CollectionFactory that neatly packages generic Collections of which can be referenced anywhere within the server. A Schema and Statement builder manages the creation and access of tables within the database which are referenced within the generic Collection. Variables used in these builders are defined using Annotations in each Model so that domain specifics are coupled together.

swing package

This package holds the logic for rendering the dialog box to display a message to the user.

utils package

This holds utility methods and classes that are used throughout the three projects.

Server

The server package holds the Route and socket handler. The Route manages the Request and uses it to act upon the given Actions. All errors are handled here which sends contextual responses to the user. The socket handler handles a given socket connection and responds based on the result given by the Route handler.

controller package

This package holds all the Controllers to be bound to the router. These Controllers hold numerous Actions within them which handle the CRUD-like operations for the various Models. Notably, the classes within these are the last Action on a specific Route that generally return the data the user had initially asked for.

middleware package

This package holds all the authentication and permission middleware for the router. These are also Actions yet are chained to run before the controllers to ensure the user making the Requests is authenticated or has the necessary permission to do so. Authentication and Permission middleware are chained on the router to protect the necessary actions from being accessed by Users that aren’t permitted or authorized to access them.

services package

The services package are Singleton services that are used through the controllers and middleware. This includes the DataService which holds the database connection and the TokenService which manages the User Sessions. Users can login through a route given they provide a valid username and hashed password pair. This stores a Session that keeps track of their token, User details and permissions.

Viewer

The viewer package holds the classes which contain the logic necessary for the Viewer to fit the specification. This includes selective rendering and sizing as well as custom action listeners.

components package

The components package holds the Message, Picture and Information classes. These classes handle the GUI for each section of a Billboard.

Test-Driven Development

Explaining how you used test-driven development and which classes were created using it.

The team utilised strict test-driven development ideology where possible to manage the continuous integration throughout the project's life. JUnit 5 unit tests are used comprehensively through the non-GUI codebase, testing multiple cases for each function to ensure rigidity. This enabled quicker bug pinching so more focus could be put on rapid expansion of the codebase.

Despite a few massive restructures which affected a majority of the core components in the Server and meant several tests became redundant along the timeline, the team still attempted to stick to a TDD mindset by building new tests where required.

Generics were utilised to make the testing of classes much simpler, as string literals and basic data types could be asserted as opposed to complex objects.

Tests are located in the test package of the java directory. These test classes follow the same structure as the main package, as tests are located on the same path as the class they test.

Each test runs as per expected which results in a more consistent and reliable software produced.

Setup + Installation

Necessary setup/installation instructions, including how to create the network Properties file, the default user's username and password and anything else you feel the client corporation/your marker will need to properly assess your software

To set up the system, a *network.props* and *db.props* is required in the root directory of the project (See Figure 1 + 2 for reference). The configuration shown in Figures 1 and 2 are as used throughout the development of this assignment. After these files have been configured, a user should now be able to run the server package Main.java which will start the server and generate a database called cab302.db in the root directory of the project.

The default admin login is username: *admin*, password: *admin*.

Figure 1: network.props

```
server.port=12345
server.address=0.0.0.0
```

Figure 2: db.props

```
jdbc.url=jdbc:sqlite:cab302.db
jdbc.schema=cab302
jdbc.username=
jdbc.password=
```

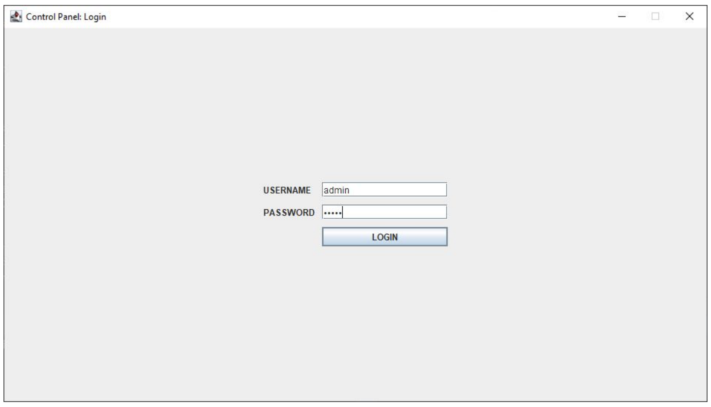
The entry point of each of the three client, server and viewer packages is the Main.java located in the root of each package.

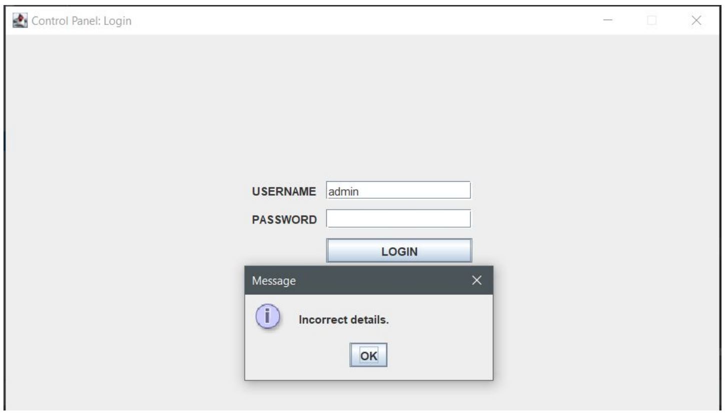
System Walkthrough

A walkthrough of your system, describing how to set up and test each part of it.

Login

Upon starting the client.Main.java, the user is prompted with the login frame. The default admin credentials are username: admin, password: admin. This will log the user into the control panel. Other credentials will work, provided they have been added in the control panel. Unsuccessful logins will display a dialog to the user that the login has been unsuccessful. If the Client can't make a connection, an appropriate message is displayed.





Incorrect details.

Main menu

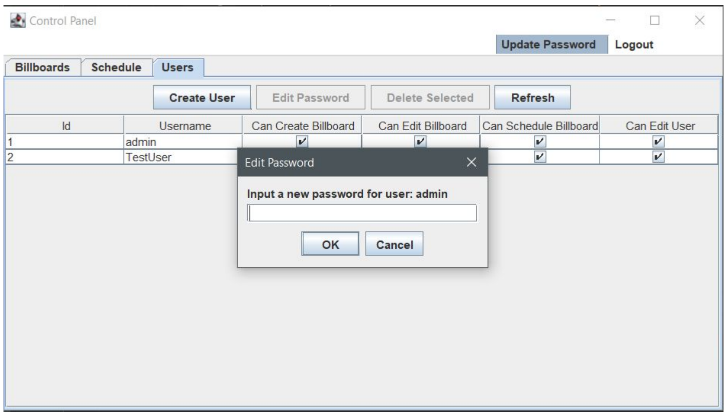
This menu holds all necessary components for the user to interact with the server. The various panels can be navigated along the top left corner. Helper buttons are located on the top right corner and can be actioned upon in any panel.

Update password

The user can update their password by clicking the update password menu. This spawns a dialog which prompts the user to enter a new password.

Logout

The user can log out at any time by clicking the logout button. This sends a request to the server to tell it to remove the Session.

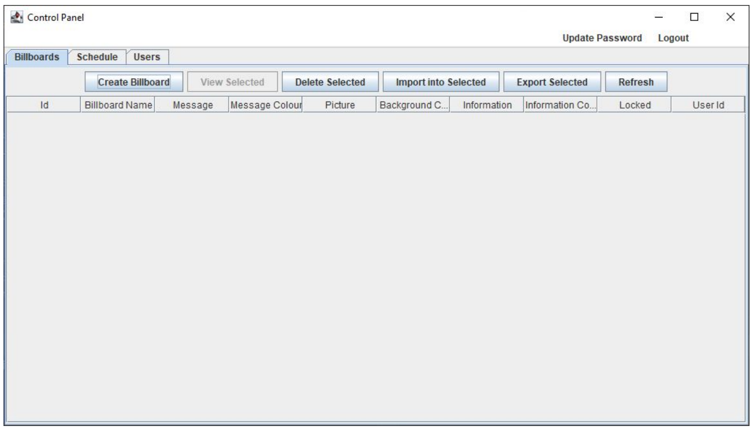


Id	Username	Can Create Billboard	Can Edit Billboard	Can Schedule Billboard	Can Edit User
1	admin	✓	✓	✓	✓
2	TestUser			✓	✓

Input a new password for user: admin

Billboard panel

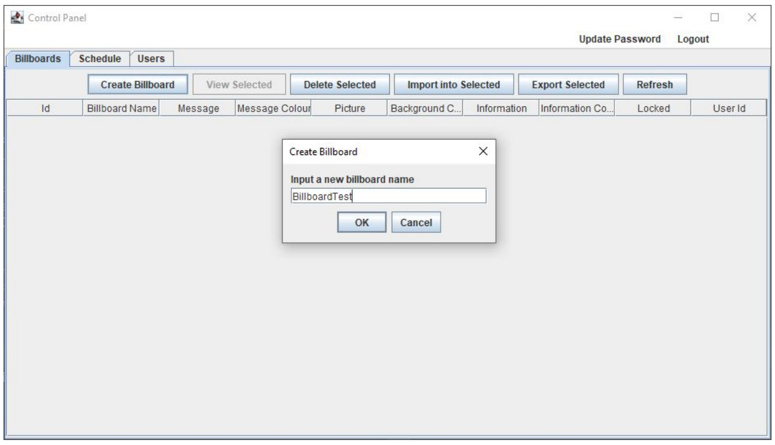
The billboard panel holds all functionality for interacting with Billboard models.



Id	Billboard Name	Message	Message Colour	Picture	Background C...	Information	Information Co...	Locked	User Id

Billboard create

The user can create a Billboard provided they have the ‘canCreateBillboard’ permission. A dialog shows up to prompt the user to input the new Billboards name.



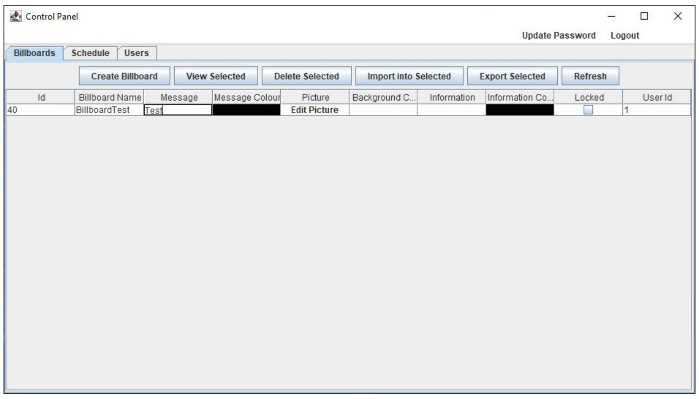
Id	Billboard Name	Message	Message Colour	Picture	Background C...	Information	Information Co...	Locked	User Id

Input a new billboard name

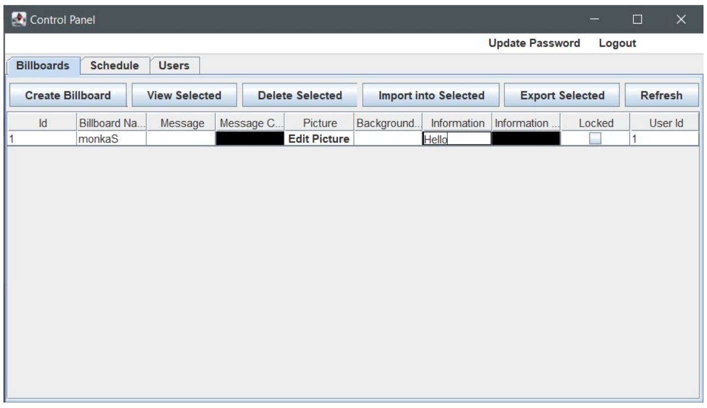
BillboardTest

Billboard edit

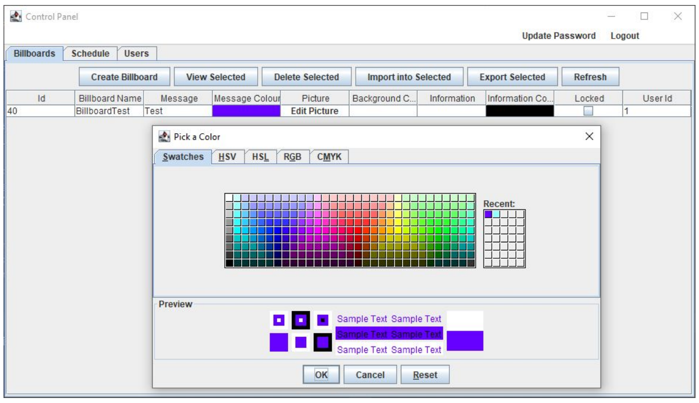
The user can edit a Billboard by editing the values in the cells. Custom cell editors have been made for managing the text colours and the picture.



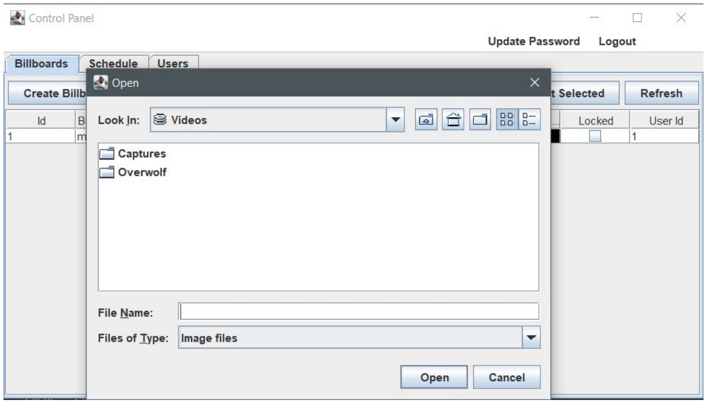
Id	Billboard Name	Message	Message Colour	Picture	Background C...	Information	Information Co...	Locked	User Id
40	BillboardTest	Test	⬛	Edit Picture			⬛	☐	1

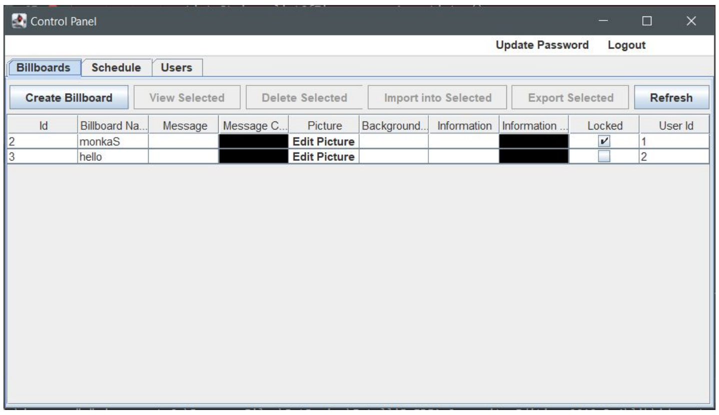


Id	Billboard Na...	Message	Message C...	Picture	Background...	Information	Information ...	Locked	User Id
1	monkaS		⬛	Edit Picture		Hello	⬛	☐	1



Id	Billboard Name	Message	Message Colour	Picture	Background C...	Information	Information Co...	Locked	User Id
40	BillboardTest	Test	🟪	Edit Picture			⬛	☐	1

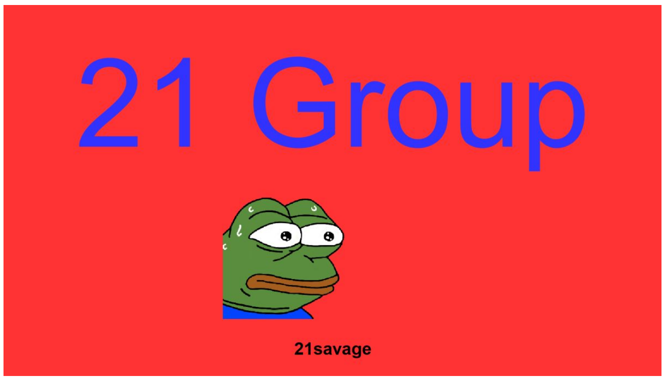


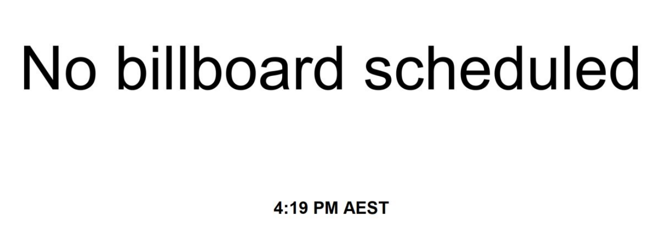


Id	Billboard Na...	Message	Message C...	Picture	Background...	Information	Information ...	Locked	User Id
2	monkaS		⬛	Edit Picture			⬛	☑	1
3	hello		⬛	Edit Picture			⬛	☐	2

Billboard View / Viewer

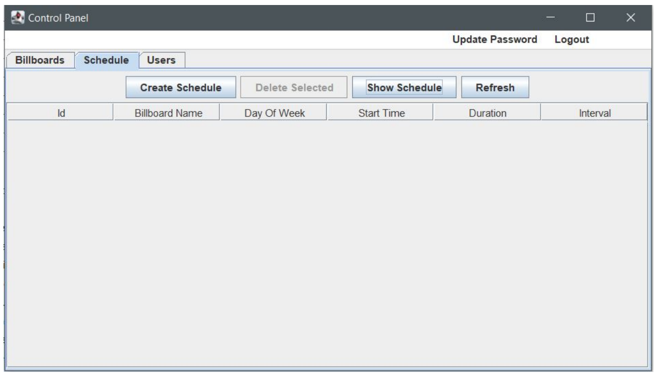
Clicking on the “View selected”, while selecting a Billboard, presents the user with a similar screen as the Viewer. The same interactions can be made with this as specified for the Viewer.





Schedule panel

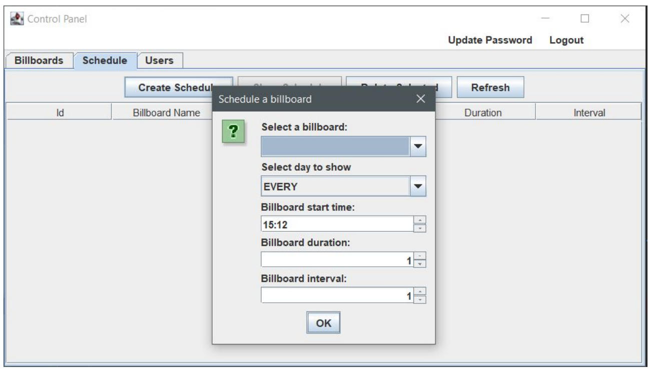
The schedule panel allows the user to schedule billboards at various times, as well as remove them and view the schedule for the current week. Given that the current user has the ‘canScheduleBillboard’ permission, this tab will show.



Id	Billboard Name	Day Of Week	Start Time	Duration	Interval

Schedule create

Creating a Schedule by clicking the “Create Schedule” button prompts the user to select the values necessary to create a Schedule. This includes selecting a: Billboard to display; day to show on (or every day); time to show at; duration to show for; and the interval to show on.



Select a billboard:

Select day to show: EVERY

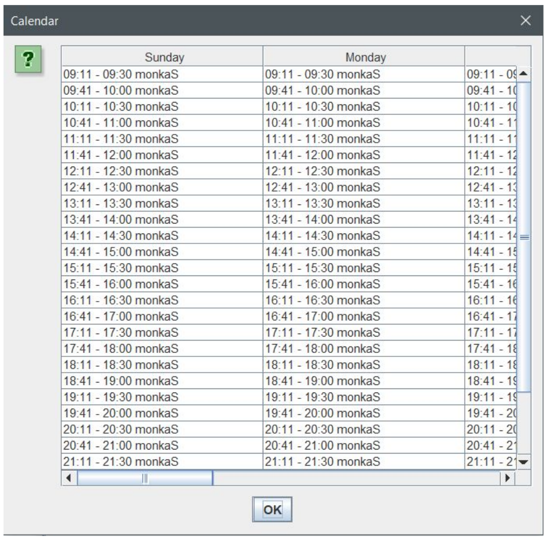
Billboard start time: 15:12

Billboard duration: 1

Billboard interval: 1

Schedule View

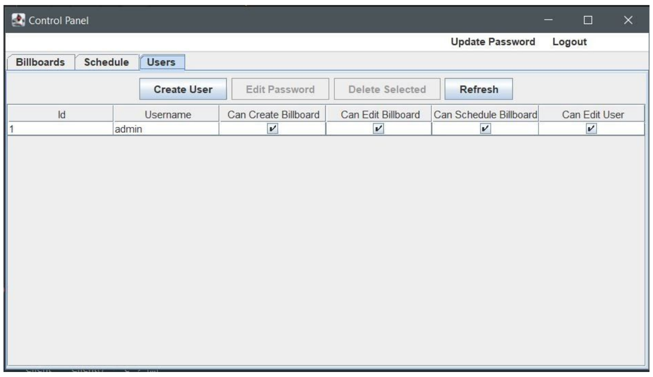
The Schedule view - accessible by clicking the “View Schedule” button - is a view of the current week's Billboard schedule. This shows a Sunday to Saturday table of each billboard viewing time from start to finish.



Sunday	Monday	
09:11 - 09:30 monkaS	09:11 - 09:30 monkaS	09:11 - 09
09:41 - 10:00 monkaS	09:41 - 10:00 monkaS	09:41 - 10
10:11 - 10:30 monkaS	10:11 - 10:30 monkaS	10:11 - 10
10:41 - 11:00 monkaS	10:41 - 11:00 monkaS	10:41 - 11
11:11 - 11:30 monkaS	11:11 - 11:30 monkaS	11:11 - 11
11:41 - 12:00 monkaS	11:41 - 12:00 monkaS	11:41 - 12
12:11 - 12:30 monkaS	12:11 - 12:30 monkaS	12:11 - 12
12:41 - 13:00 monkaS	12:41 - 13:00 monkaS	12:41 - 13
13:11 - 13:30 monkaS	13:11 - 13:30 monkaS	13:11 - 13
13:41 - 14:00 monkaS	13:41 - 14:00 monkaS	13:41 - 14
14:11 - 14:30 monkaS	14:11 - 14:30 monkaS	14:11 - 14
14:41 - 15:00 monkaS	14:41 - 15:00 monkaS	14:41 - 15
15:11 - 15:30 monkaS	15:11 - 15:30 monkaS	15:11 - 15
15:41 - 16:00 monkaS	15:41 - 16:00 monkaS	15:41 - 16
16:11 - 16:30 monkaS	16:11 - 16:30 monkaS	16:11 - 16
16:41 - 17:00 monkaS	16:41 - 17:00 monkaS	16:41 - 17
17:11 - 17:30 monkaS	17:11 - 17:30 monkaS	17:11 - 17
17:41 - 18:00 monkaS	17:41 - 18:00 monkaS	17:41 - 18
18:11 - 18:30 monkaS	18:11 - 18:30 monkaS	18:11 - 18
18:41 - 19:00 monkaS	18:41 - 19:00 monkaS	18:41 - 19
19:11 - 19:30 monkaS	19:11 - 19:30 monkaS	19:11 - 19
19:41 - 20:00 monkaS	19:41 - 20:00 monkaS	19:41 - 20
20:11 - 20:30 monkaS	20:11 - 20:30 monkaS	20:11 - 20
20:41 - 21:00 monkaS	20:41 - 21:00 monkaS	20:41 - 21
21:11 - 21:30 monkaS	21:11 - 21:30 monkaS	21:11 - 21

User panel

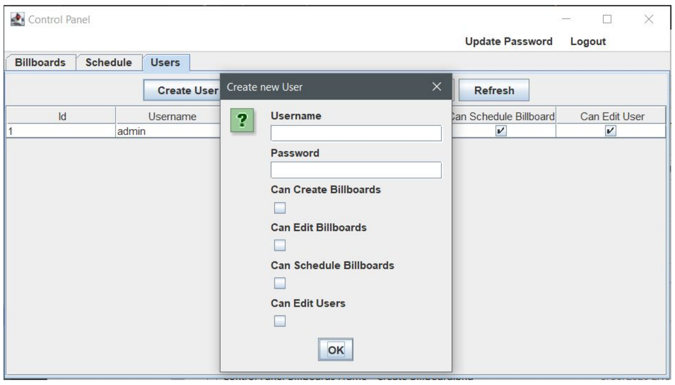
The user panel is where parent user accounts can edit child Users. Provided that they have the ‘canEditUser’ permission, this tab will show. Users can be created, edited and deleted from this panel.



Id	Username	Can Create Billboard	Can Edit Billboard	Can Schedule Billboard	Can Edit User
1	admin	✓	✓	✓	✓

Create new user

Upon clicking ‘Create User’, a dialog will spawn that prompts the user to create a new User. This includes the username, password and a checkbox list of all the permissions tied to that User.



Username

Password

Can Create Billboards ☐

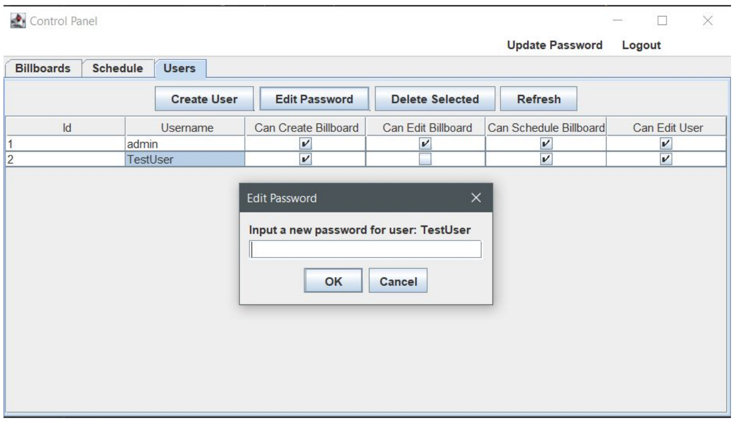
Can Edit Billboards ☐

Can Schedule Billboards ☐

Can Edit Users ☐

Edit user

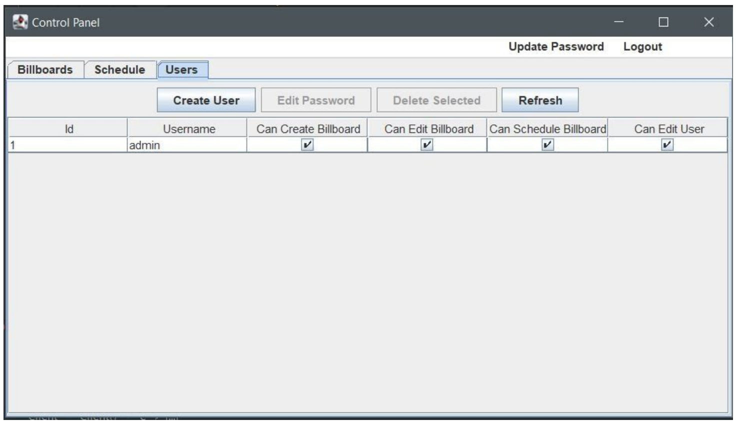
The user can edit the password of a selected user and edit their permissions by clicking on the respective checkbox cells.



Id	Username	Can Create Billboard	Can Edit Billboard	Can Schedule Billboard	Can Edit User
1	admin	✓	✓	✓	✓
2	TestUser	✓		✓	✓

Edit Password

Input a new password for user: TestUser



Id	Username	Can Create Billboard	Can Edit Billboard	Can Schedule Billboard	Can Edit User
1	admin	✓	✓	✓	✓