TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Mikhail Polomoshnov 185161IADB

Line of Business Front-end Application Migration on the Example of Playtech PLC

Bachelor's thesis

Supervisor: German Mumma
Master of Science

Tallinn 2025

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mikhail Polomoshnov 185161IADB

Ärikriitilise eesrakenduse migreerimine Playtech PLC näitel

Bakalaureusetöö

Juhendaja: German Mumma
Magister

Tallinn 2025

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Mikhail Polomoshnov

01.01.2025

Abstract

Organisations across the globe design and implement custom software solutions to fulfil their unique needs. Over time, technologies used to develop proprietary systems become obsolete. This challenges organisations with the task of migrating legacy applications to more modern technologies. The author's organisation was faced with a similar challenge when the framework used to develop a front-end application for a project reporting system reached its end of life. Therefore, the main objective of this work was to formulate an application-specific migration approach and rewrite the legacy front-end application. To achieve this goal, the author utilised a variety of methods, including analysis, prototyping, testing, development and deployment.

In the analysis part, the author examined the old application and compared common technologies and migration strategies that could be utilised to solve the problem of the work. As a result, Angular was selected as the main framework for the new solution, and an incremental rewrite with two separate applications was chosen as the migration approach. Then, the author developed and validated a prototype for the final solution. Subsequently, the author analysed functional and non-functional requirements for the new application. Lastly, in the implementation part of the work, the author covered the development, testing, and deployment of the new solution.

As a result of this work, the main objective has been achieved. The new application has been developed and deployed to the production environment. The new solution is implemented using state-of-the-art front-end technologies and standard programming approaches, providing a comprehensive solution to the main problem of this work. The new solution demonstrates major improvements over the legacy application across all key areas. The new application provides a superior experience to the users and is well-positioned for continued development and future growth.

This thesis is written in English and is 38 pages long, including 6 chapters, 12 figures and 4 tables.

Annotatsioon
Ärikriitilise veebipõhise eesrakenduse migreerimine Playtech PLC näitel

Ettevõtted üle maailma arendavad kohandatud tarkvaralahendusi, et täita oma unikaalseid vajadusi. Aja jooksul ettevõte süsteemide arendamiseks kasutatud tehnoloogiad vananevad. See protsess esitab ettevõtetele väljakutse viia oma rakendused üle kaasaegsetele tehnoloogiatele. Autori organisatsioon seisis silmitsi sarnase väljakutsega, kui projektide aruandlussüsteemi veebipõhise eesrakenduse arendamiseks kasutatud raamistik jõudis oma eluea lõpuni. Käesoleva töö põhieesmärgiks oli defineerida eesrakenduse migratsiooni lähenemisviisi ning kirjutada eesrakendust ümber. Töö eesmärgi saavutamiseks kasutas autor erinevaid meetodeid, sealhulgas analüüsi, prototüüpimist, testimist, arendamist ja juurutamist.

Analüüsiosas oli uuritud vana rakendus ning omavahel võrreldud kaasaegsed tehnoloogiad ja migratsiooni lähenemisviisid, mida saaks kasutada töö eesmärgi saavutamiseks. Uue lahenduse põhiraamistikuks oli valitud Angular. Migratsioonimeetodiks valiti samm-sammuline ümberkirjutamine kahe eraldi rakendusega. Seejärel, autori poolt oli välja töötatud lahenduse prototüüp. Järgmisena oli analüüsitud uue rakenduse funktsionaalsed ja mittefunktsionaalsed nõuded. Töö praktilises osas tegeles autor uue lahenduse arendamise, testimise ja juurutamisega.

Käesoleva töö tulemusena said põhieesmärgid täielikult täidetud. Uus lahendus on arendatud, paigaldatud toodangusse ja võetud kasutusele. Uus lahendus on loodud kasutades kaasaegseid tehnoloogiaid ja standardseid programmeerimisvõtteid ning pakub põhjalikku lahendust käesoleva töö põhiprobleemile. Uus lahendus demonstreerib olulisi täiustusi võrreldes vana rakendusega kõigis võtmevaldkondades. Uus rakendus pakub kasutajatele paremat kasutuskogemust ning on hästi valmistatud tulevaseks arenguks.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 38 leheküljel, 6 peatükki, 12 joonist, 4 tabelit.

List of abbreviations and terms

AI	Artificial Intelligence
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CLI	Command Line Interface
CSS	Cascading Style Sheets
DI	Dependency Injection
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IIS	Internet Information Services
LTS	Long-term Support
MIS	Management Information System
NGRX	Angular Reactive Extensions
PLC	Public Limited Company
RXJS	Reactive Extensions for JavaScript
SASS	Syntactically Awesome Style Sheets
SPA	Single-Page Application
UI	User Interface
UX	User Experience
URL	Uniform Resource Locator
VM	Virtual Machine
XML	Extensible Markup Language

Table of contents

List of figures

List of tables

1 Introduction

Organisations across the globe develop custom software solutions to address their unique needs. Over time, technologies used to develop proprietary systems become obsolete. Over the last few years, countless organisations have been faced with the challenge of migrating their AngularJS applications to more modern technologies to mitigate various risks. The author's organisation faced the same problem with a legacy front-end application for a project reporting system. The main objective of this work is to formulate an application-specific migration approach and rewrite the legacy front-end application.

In the background and motivation part of the work, the author will introduce the organisation and the legacy system. The author will also introduce the problem, objective, methodology, scope, limitations and relevance of the work.

In the analysis part, the author will examine the old application. Then, common technological solutions and migration strategies will be thoroughly analysed in the context of the project reporting system. After that, the author will develop and validate a prototype for a possible solution. Subsequently, functional and non-functional requirements for the new solution will be covered. As a result of the analysis, the author will be able to select the main framework and migration approach for the new solution as well as understand the requirements for it.

In the practical part of the work, the author will describe the process of implementing the new solution. The implementation part will cover the work process organisation, low-level technological decisions, and the process of development, testing and deployment.

As a result of this work, a new front-end application will be developed and deployed to the production environment.

The following work is written in English to comply with the author's organisation requirements.

2 Background and motivation

In the following chapter, the author will introduce key concepts crucial for the understanding of the thesis. After that, the author will present the company and the system background. Subsequently, the author will define the problem and the main goal of the thesis. Finally, the scope, limitations, methodology and relevance of the work will be covered.

2.1 Front-end application

A web client application, also known as a front-end application or browser-based application, is a part of an application system that the user directly interacts with using a web browser. It is responsible for rendering the user interface, enabling user interactions, and handling communication with other parts of the system. It is hosted on a remote server and accessed through a web browser, which means it does not need to be permanently stored or installed locally [1]. It encompasses all the graphical elements that users interact with, e.g. forms, buttons, tables, graphs, and input fields [2].

Typically, a front-end application is powered by 3 main web technologies: HTML, CSS and JavaScript. HTML is a markup language used to create and structure the content of a front-end application. CSS is a design language that specifies the presentation and styling of the content defined by the HTML. JavaScript is a multi-purpose programming language. In the context of front-end development, JavaScript powers the underlying logic of an application. It is used to dynamically create and update the content, create animations, and communicate with other parts of the system [3].

2.2 Web client application framework

In software engineering, a front-end framework is a collection of systematically organized, pre-written, reusable software modules designed to facilitate a more efficient development of front-end applications. Typically, a front-end framework provides an engineer with pre-written components, design patterns, architectural approaches, tools, specifications and standards [4].

The main purpose of using a front-end framework is to enhance the application development and maintenance speed by eliminating the need to define standards and write base-level code [4]. Essentially, a framework provides a foundation for developing applications, so the developers can focus on implementing the business logic of an application.

2.3 Organisation background

The author of the following thesis is employed by Playtech PLC – an international enterprise which operates in the gambling industry. Playtech was founded in 1999 and is listed on the London Stock Exchange. Headquartered in the UK, Playtech has offices in 20 countries with around 7800 employees [5].

Playtech specialises in developing platforms, content and services for private and business customers. With its B2B offerings, Playtech strives to be a technology partner of choice for betting and gambling operators in regulated markets [5].

The author's business unit is responsible for the management, development and maintenance of the internally used applications. The author's team, in particular, is focused on engineering and maintaining custom solutions that fulfil the ever-changing needs of the organisation.

2.4 System background

The author's organisation utilises an in-house built software system for project management, budgeting, and reporting purposes. The system is used by a wide range of employees and managers across the organisation with up to 7500 total users. The number of daily users normally falls between a few tens and a few hundreds.

The system is a typical example of a MIS or management information system. It aggregates internal operational data (e.g. project data, employee work log data, project cost and time estimations, etc) from a number of source systems or directly from the employees. Using the collected data, it provides the users with scheduled reports, on-demand reports as well as UI for data processing and analysis. The main stakeholders of this system are the budgeting team employees who use it in their daily work to compose reports, coordinate the project budgeting, and analyse the project and employee data.

The system is composed of a database, a server-side application, a front-end application, and a number of scheduled console applications responsible for integrations with other systems across the organisation (see Figure 1).

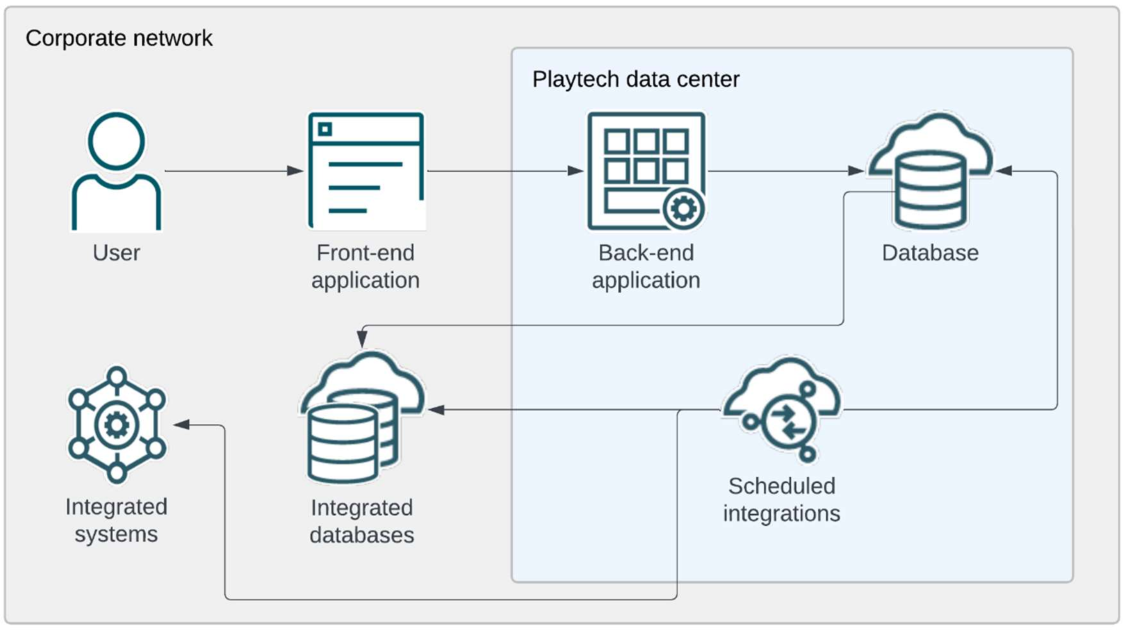


Figure 1. High-level system scheme.

The author's team fully owns the system, and the author is the sole engineer developing and maintaining it. The application system has been in use since 2017 and has been developed since 2016 using the state-of-the-art frameworks of that period: AngularJS and .NET Framework. In the context of this work, the author will predominantly focus on the front-end part of the system.

2.5 Problem statement

The front-end application for the aforementioned project reporting system has been built using the AngularJS JavaScript framework (also known as Angular 1) which was released in 2010 and reached its end-of-support date on the 1st of January 2022 [6].

Persisting with an obsolete framework poses various risks to the organisation. The main risks include [7]-[9]:

- Security vulnerabilities
- Insufficient performance
- Browser incompatibility
- Decreased productivity of users and developers
- Lack of access to new tooling and libraries
- Lack of support for the used tooling and libraries
- Small talent pool and unsatisfactory developer experience

The are several other problems specifically associated with the project reporting front-end application, which will be discussed in more detail in the following chapters. The main issues include:

- Bugs
- Dead code (i.e. code that is unnecessary for successful operation of the application)
- Obsolete features
- Performance issues
- Maintainability issues
- Outdated user interface
- Lack of user input validation

2.6 Goal statement

The main objective of this work is to formulate an application-specific migration approach and rewrite the legacy front-end application.

As a result of this work, a new front-end application will be developed and deployed to the production environment.

2.7 Methodology

The author will utilise the following methods to achieve the goal:

- Problem analysis
- Common solutions analysis
- Prototyping

- Functional and non-functional requirements analysis
- Development
- Testing
- Deployment

2.8 Scope and limitations

The thesis will cover the analysis and definition of the application-specific migration approach as well as the process of rewriting and deploying the new application. In the scope of the work, the author will implement the base logic of the new application (e.g. state management, user management, base services, routing) and will fully migrate four views from the old application. Transfer of all thirty to forty views to the new application would require an extended time and falls outside of the scope of the thesis.

Due to internal security compliance requirements, the topics of hosting infrastructure, authentication and authorization will be covered in a general manner in the thesis.

While the author is migrating both the front-end application and the server-side application at the same time, the server-side application will not be covered in this work.

2.9 Relevance

Over the last few years, countless organisations have been faced with the challenge of migrating their AngularJS applications to more modern technologies to mitigate various risks. Hence, the problem the author aims to tackle is relevant. To solve the problem, the author will utilise state-of-the-art front-end technologies for enterprise applications.

Many organisations use project reporting software to plan, organise and monitor the work. While there are several options available on the market, custom solutions remain relevant, especially for large organisations that can allocate resources to develop them. Custom solutions are created to address the specific needs of an organisation that cannot be satisfied by a generic solution from the market. The stakeholders within an organisation can fully define the functionality of a custom solution to ensure that it solves their problems the best. The same cannot be said about the off-the-shelf solutions.

3 Analysis

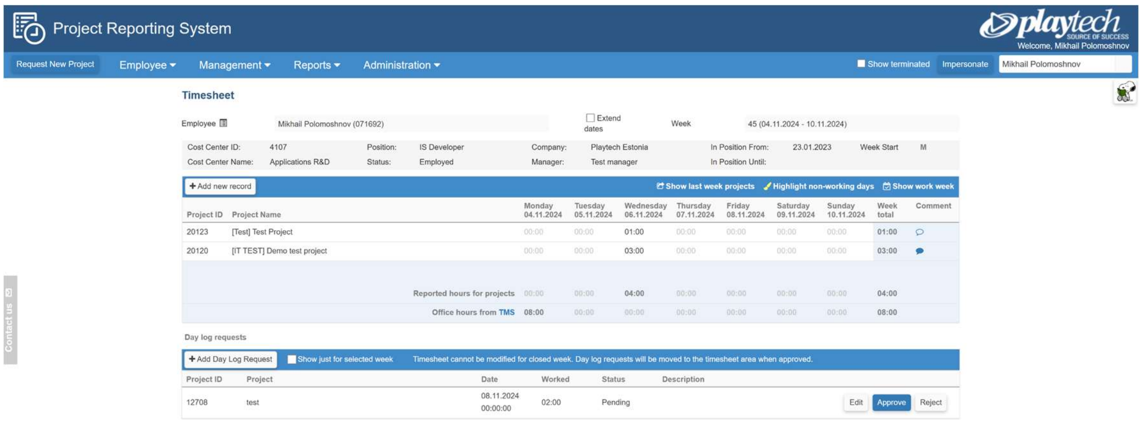
In the following chapter, the author will first analyse the current solution from the technical point of view. The analysis will cover currently used technologies, architectural approaches, design patterns, issues and limitations. Then, the author will analyse and compare different technologies and migration approaches that could be utilised to implement a new solution. Next, the author will cover the prototyping. Lastly, the author will analyse the functional and non-functional requirements for the new solution.

3.1 Technical analysis of the current solution

The current solution implements the single-page application design pattern, which means that the application dynamically rewrites the content on a single web document instead of loading a new web document on each interaction [4]. The application is built using a free and open-source front-end framework called AngularJS. AngularJS is based on three fundamental web technologies: HTML, CSS and JavaScript. AngularJS reached its end-of-support date and does not receive any official updates or bug fixes anymore. The framework has several known security vulnerabilities and bugs [7]. The framework is considered to be less performant than its modern counterparts [10]. Lastly, the framework community size is decreasing day by day [8].

The application code is also written in JavaScript, which means that it does not support strong typing. This makes the development and troubleshooting processes more challenging [9]. In addition, JavaScript is not supported by some of the useful front-end development tools that require strong typing.

To facilitate the development process and provide a consistent user interface across the application, a UI library called Kendo UI is employed. The UI library is based on jQuery – a popular JavaScript library designed to simplify common front-end development tasks [11]. The UI library also makes use of the AngularJS Directives which will be discussed in more detail later in this section. While the UI library provides a wide selection of UI components, it lacks many of the modern features and appears dated (see Figure 2). Any drastic updates to the UI of the project reporting application are complicated by the fact that the UI library reached its end-of-support date. The library does not contain many modern features and will not receive any updates or bug fixes in the future.



Timesheet

Employee	Mikhail Polomoshnov (071692)	Extend dates	Week 45 (04.11.2024 - 10.11.2024)

Cost Center ID:	4107	Position:	IS Developer	Company:	Playtech Estonia	In Position From:	23.01.2023	Week Start	M
Cost Center Name:	Applications R&D	Status:	Employed	Manager:	Test manager	In Position Until:			

+ Add new record | Show last week projects | Highlight non-working days | Show work week

Project ID	Project Name	Monday 04.11.2024	Tuesday 05.11.2024	Wednesday 06.11.2024	Thursday 07.11.2024	Friday 08.11.2024	Saturday 09.11.2024	Sunday 10.11.2024	Week total	Comment
20123	[Test] Test Project	00:00	00:00	01:00	00:00	00:00	00:00	00:00	01:00	
20120	[IT TEST] Demo test project	00:00	00:00	03:00	00:00	00:00	00:00	00:00	03:00	
	Reported hours for projects	00:00	00:00	04:00	00:00	00:00	00:00	00:00	04:00	
	Office hours from TMS	08:00	00:00	00:00	00:00	00:00	00:00	00:00	08:00	

Day log requests

+ Add Day Log Request | Show just for selected week | Timesheet cannot be modified for closed week. Day log requests will be moved to the timesheet area when approved.

Project ID	Project	Date	Worked	Status	Description	
12708	test	08.11.2024 00:00:00	02:00	Pending		Edit Approve Reject

Figure 2. Old project reporting system. Timesheet page.

The application is composed of approximately forty unique views and corresponding controllers. It has four distinct user groups: regular employees, managers, budgeting team employees and administrators. The groups can use different views that can be accessed from the navigation bar. The views shared between different groups expose different functionality for different users based on the group.

The current solution follows a controller-based architectural approach as opposed to a more modern component-based approach. At the foundation of controller-based AngularJS application are five key concepts: modules, controllers, directives, templates and services (see Figure 3). A module is a container file that encapsulates and wires together different parts of the application including other modules. The root module defines the application and acts as an entry point for it. A service is an injectable object that encapsulates commonly used code and performs one specific task. A template is a part of the AngularJS application that defines the content rendered in the browser. A directive is a custom-made or built-in marker on a DOM element that transforms the element or attaches some specific behaviour to it. Finally, the controller is an object encapsulating the logic that powers a specific template. The controller and the template interact with each other using a scope object and different binding options provided by the framework. The scope is an object that encapsulates the controller's properties and methods that can be accessed by the template. [12]

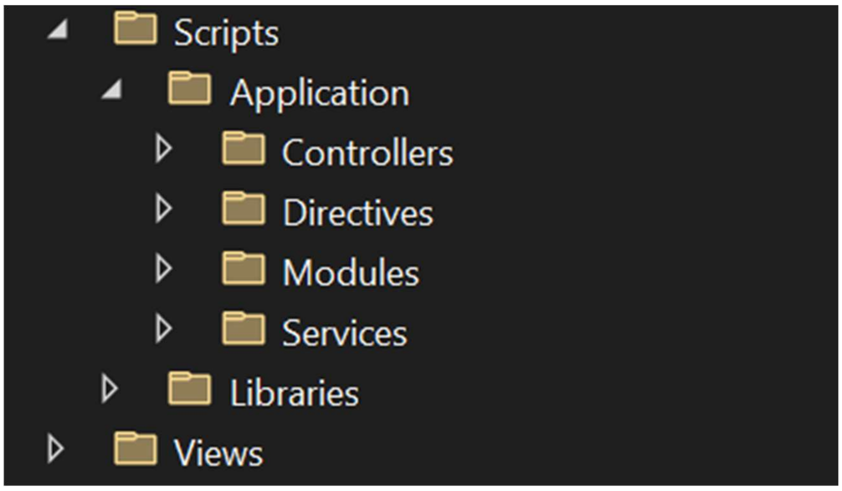


Figure 3. Controller-based AngularJS project structure

The application routing is implemented in the root application module using the ngRoute AngularJS module. In the application navigation bar, the users can only see the routes accessible to their user group.

The application communicates with the server-side application via HTTP requests utilising AJAX techniques. By convention, HTTP requests in the AngularJS application should be encapsulated in separate services. In the current solution, the HTTPS requests are implemented within the controllers which is considered to be an antipattern [13]. This approach does not follow the separation of concerns principle and promotes code duplication.

The application does not utilise any libraries or custom solutions for managing its global state. This means that commonly used and rarely changed data is not shared between different parts of the application and needs to be fetched from the server-side application each time the user navigates from one page to another. This negatively affects the user experience and overall performance of the system [14].

The application is served by an IIS web server installed on a Windows Server virtual machine. The virtual machine itself is hosted on an on-premises server. The web client application is hosted on the same VM and IIS site as the server-side application.

The front-end application is secured behind a firewall and can only be accessed from the Playtech corporate network. The IIS site, and consequently the application, prohibits anonymous authentication and can only be accessed by internal users with an active Microsoft Entra account. The application supports single sign-on and uses username and password authentication. Authentication is based on a widely adopted computer-network authentication protocol and is handled by the application's IIS web server. Authentication is session-based, meaning that the user session data is stored on the server, and the client application needs to provide a session identifier on each request to the server for authentication. The application uses HTTPS, ensuring secure data transmission.

The project reporting application contains several unfixed bugs. On top of that, its codebase contains many examples of dead code, code repetition, "magic" values and other common programming antipatterns. Furthermore, the application has insufficient user input validation in some views. The application load speed and overall performance are mediocre and could be improved. All these problems result in a deteriorated user experience as well as more complex development and maintenance processes.

Furthermore, the application contains several obsolete features that are no longer used. At the same time, the application is missing some features that have been long awaited by the users. This means that the business logic of the application needs to be reexamined, and redundant features must be removed from the application.

3.2 Analysis of common technological solutions

Selecting an appropriate technological solution is a crucial step in mitigating the risks associated with the old application. One possible technological solution to the problem of this work could be to adopt a modern front-end framework and migrate the old application. Another technological solution could be not to rewrite the application but to utilise a commercial long-term support solution for the old framework. In this section, the author will analyse and compare different technological solutions in the context of the project reporting application migration.

3.2.1 Commercial long-term support

The simplest way to handle framework end of life is not to migrate and continue using the old technology. While persisting with the last official AngularJS release is undesirable as it poses too many risks for the organisation, commercial long-term support offerings from third-party vendors might seem more appealing.

Commercial extended support solutions for AngularJS are offered by two software development companies: HeroDevs and OpenLogic. Both solutions provide similar functionality and mitigate two major risks associated with the unsupported front-end framework: security vulnerabilities and browser incompatibilities. Additionally, the LTS solutions provide stability updates that might benefit some applications. [7], [15]

Commercial LTS solutions for AngularJS have several strong advantages. They enhance the longevity, security and stability of the application by eliminating two main risks of the unsupported framework. Setting up such a solution for an existing application would require little effort from the developer [7]. However, this approach does not address some critical issues associated with the old framework. In particular, the AngularJS would generally have worse performance than a typical modern framework [10]. The old framework lacks some modularity and code reusability features [8]. The outdated framework lacks some modern features and tooling [8]. The framework and its tooling have limited community support and will not receive any updates. Furthermore, continuing with the old framework would mean that the application design issues will remain unsolved. The LTS solution will not address the problem of a limited pool of skilled talent and unsatisfactory developer experience. The organisation will need to perform a thorough security and compliance check on the third-party vendors, which would require additional time investment. The LTS solutions are not free and will mean additional costs to the organisation. Lastly, the long-term support for AngularJS in the future, so the rewrite remains unavoidable.

In essence, proceeding with an LTS solution for the existing application will only delay the inevitable migration to a newer technology. At the same time, this approach will not solve several key issues of the old solution and will introduce additional costs to the organisation. For these reasons, it was ultimately decided not to proceed with this approach.

3.2.2 Modern front-end frameworks for enterprise applications

The front-end technology landscape is abundant with different frameworks. Every year a few promising frameworks emerge, growing the number of options for the developers, while, at the same time, making the choice more difficult [16]. In the scope of this work, the author will develop an enterprise application, which implies that the underlying framework must meet a set of strong requirements. In particular, the target framework must be performant, reliable and scalable. It must have a large talent pool, a big community, reputable developers and extensive documentation. It must receive regular updates, and it must have an abundance of features and tooling to satisfy the evolving needs of the organisation**.**

By and large, three frameworks meet all of the aforementioned requirements and are considered to be well-suited for modern enterprise front-end applications: Angular, React and Vue. These frameworks provide the best balance of reliability, performance, popularity, tooling and support. Therefore, the author only considered these three frameworks for the new application. Next, the author will analyse and compare Angular, React and Vue JavaScript frameworks in more detail (see Table 1) [4], [16]-[19]:

Table 1. Comparison of most common enterprise front-end frameworks.

Criteria	**Angular**	**React**	**Vue.js**
Technology type	Comprehensive all-in-one framework	Library; Mostly concerned with the user interface part of an application	Comprehensive all-in-one framework
Developers	Google	Facebook	Reputable open-source developers
Architectural approach	Component-based architecture	Component-based architecture	Component-based architecture
Scalability	Great scalability; Often regarded as the standard for large-scale enterprise applications	Good scalability; Considered a good option for medium and large enterprise applications	Generally considered less scalable than the other two; Best for small and medium-sized applications
Performance	High; Generally considered slightly less performant	High; Generally considered slightly more performant than others	High; Generally considered slightly more performant than Angular
Rendering	Supports both client-side and server-side rendering	Supports both client-side and server-side rendering	Supports both client-side and server-side rendering
DOM type	Real DOM	Virtual DOM	Virtual DOM
Data binding	Supports one-way binding and two-way binding	Only supports one-way binding out of the box	Supports one-way binding and two-way binding

Criteria	**Angular**	**React**	**Vue.js**
State management	Wide range of third-party state management options	Wide range of third-party state management options	Wide range of official and third-party state management options
Language support	TypeScript and JavaScript	TypeScript and JavaScript	TypeScript and JavaScript
Cost and source code availability	Free and open-source framework	Free and open-source framework	Free and open-source framework
Community size	Large community	The largest community	Smaller community
Documentation	Extensive official and community documentation	Extensive official and community documentation	Extensive official and community documentation
Tooling	Abundant with open-source and commercial tooling	Abundant with open-source and commercial tooling	Abundant with open-source and commercial tooling
Learning complexity	The most complex	Moderate	The least complex
Talent pool	Large	Large	Small
Tooling for migration from AngularJS	Official and third-party tooling for code conversion and "hybrid" application development	Third-party tooling for code conversion and "hybrid" application development	Third-party tooling for code conversion and "hybrid" application development

All compared technologies possess the features required for modern enterprise front-end applications. Each of them would be a great option for developing a medium-sized application like the one this work is concerned with.

There are two additional aspects to consider when selecting a framework for a new project: the expertise within the team and common practices within the organisation. It happened, that the author of this work had previously worked on Angular applications at his former workplace before coming to Playtech and taking on this project. Therefore, choosing Angular would mean that the author would not have to spend excessive time obtaining additional qualifications for this project. Furthermore, Angular has been the front-end technology of choice within the author's business unit for many years due to its vast ecosystem, strong backing and opinionated design (i.e. abundance of official tooling and default approaches). Therefore, most of the front-end solutions in the author's business unit have been developed using Angular, and most of the author’s colleagues have expertise in it. These factors ultimately played the biggest role in the decision process, making the author pick Angular for the new project reporting application.

3.3 Analysis of common migration strategies

Another crucial consideration when rewriting a front-end application is the migration strategy. With respect to the migration timeline, the strategies can be divided into two categories: big bang migration (also known as synchronous migration) and gradual migration (also known as incremental or asynchronous migration) [20]. The latter category is constituted by two architectural approaches: hybrid application and separate applications. In this section, the author will analyse and compare the most common migration approaches in the context of the project reporting application.

3.3.1 Big bang migration

Big bang migration is a migration approach that is performed in one operation, i.e. the switch from the old application to a new one happens at a single point in time. This implies that the new solution must be ready to replace an old application before the migration happens. As a rule, after the migration is finished, the old application is switched off and not used anymore. The big bang migration is atomic, meaning it avoids having a continuous transitional state [20].

A big advantage of this approach over the others is that it reduces the complexity and costs of the migration associated with the simultaneous development, build and deployment of two separate applications or a hybrid application [21]. It eliminates the need for interoperability between the old and code. A major drawback of the big bang approach is that it is very resource-intensive and requires a large time investment since the entirety of the application must be developed and tested before the first deployment [20]. This might make it a bad choice for small teams or large projects. Another pitfall of this approach is that the application or some of its parts may become obsolete during the long development stage as the business needs tend to change over time. This, in turn, would mean lost opportunities and a waste of resources for the business.

3.3.2 Hybrid application migration

Hybrid application migration is a type of migration that occurs in increments over time when parts of an old application are rewritten using the new technology. This implies that parts of the application, written with different frameworks, are intertwined and work together as one [22].

From the architectural point of view, this migration approach can be split into two categories: inside-out migration (also known as bottom-up migration) and outside-in migration (also known as shell migration) [23].

With the bottom-up migration, the developer can start rewriting child components of the application, connecting them to the old root components. This approach allows developers to migrate the business logic of the application without the need to update the root components upfront. This approach, though, does not allow to immediately utilise some of the new features and benefits of the new framework [24]. Additionally, this approach implies that the application root components are updated last, meaning that all previously rewritten child components need to be updated and tested to ensure compatibility with the new root components [23].

The outside-in migration approach is an adaptation of a strangler pattern where the application root components are migrated first. After the top-level component is migrated, the legacy child components are linked to it. Then, the child components are rewritten one by one. This approach allows to leverage benefits of the new framework upfront and requires less configuration and testing in the long perspective. [24]

The need for interoperability between the old and the new code in a hybrid application means that the developers must utilise additional tooling and techniques to achieve it. When it comes to the AngularJS to Angular hybrid migration, two technologies developed by the Angular team are most commonly used: Angular elements and ngUpgrade library.

Angular elements allow the developer to package Angular code as Web Components, which is a web standard for defining HTML elements in a framework-agnostic way. With Angular elements, the developer can create isolated reusable components that exist outside of the AngularJS context. This encourages less coupling between the old and the new code. At the same time, this makes Angular elements a less suitable option for applications with heavily coupled components and complex states, as the standard Web Component communication options can be limiting. [25]

The ngUpgrade library enables running both Angular and AngularJS in the same application simultaneously [22]. It allows us to either "upgrade" AngularJS code or "downgrade" Angular code. This means that a special wrapper around the old or the new code is added ensuring compatibility with the other framework [23]. It enables close coupling between the old and the new code, which can be beneficial in some cases (e.g. migrating long, heavily coupled business processes). A big disadvantage of this library when it comes to the project reporting system migration, is that a heavy preparation of the old codebase would be required in order to fully utilise the benefits of the library (e.g. an update to AngularJS v1.6, migration to the component-based architecture, etc) [26].

The hybrid migration can also be performed in a framework-agnostic way using other technologies, for example, a different implementation of Web Components or iframe. Though they are not specialised for our kind of migration, and for that reason, they will not be analysed in this work.

The disadvantages of the hybrid migration, regardless of the architectural approach or technologies used, come from the need to use two different frameworks in the same application. This approach adds additional complexity to the codebase and requires many updates to the old code. In addition to that, hybrid applications are generally considered to be less performant due to the need to run two different frameworks at the same time. Another disadvantage of this approach is that hybrid applications have a larger build size, which increases the application loading time degrading the user experience. Finally, hybrid applications have longer build time, which hinders the development speed.

3.3.3 Gradual separate application migration

Gradual separate application migration can be characterised by a continuous transition when the old application functionality is incrementally migrated to a new application. This means that the old and the new applications run concurrently in the production environment during the migration period. An increment in the context of a front-end application could mean a single view, a process, or a set of views or processes.

This migration approach can be split into two categories by the adoption strategy: parallel and phased. The parallel adoption implies that specific functionality remains available in the old application after it has been migrated. This approach prioritises the gradual adoption of a new application and provides a backup option. In this case, the migration process is finished when the old and new applications reach feature parity. Phased adoption means that specific functionality is removed from the old application once its migration phase is finished. The removal can happen either right after migration or after some time. This approach prioritises more rapid adoption of the new solution. In this case, the migration process is finished when the new application fully assumes the functionality of the old application. [27], [28]

This approach, if properly executed, requires minimal effort for the integration between the old and the new applications. The communication between the two applications typically happens via URL forwarding and route parameters. This approach requires minimal updates to the legacy code and ensures a much simpler codebase compared to the hybrid approach. Each of the two applications has a smaller build size compared to the hybrid application which decreases the initial load time of an application. One disadvantage of this approach is that the effort required before the initial deployment of the new application is larger compared to the hybrid approach, as the new application base must be implemented from scratch. Another disadvantage of this approach is that it makes the deployment process more complex, as the two applications are deployed separately. Lastly, the lack of options for integration with the old application might make this approach unsuitable for systems with many long or codependent processes that require shared state or extensive communication between each other.

3.3.4 Migration approach selection

The big bang strategy would not be suitable for this migration, since the author is the sole developer working on this project. Due to this circumstance, the big bang approach poses too much risk and is impossible to complete within an adequate timeframe.

The project management application does not have any long processes or tightly coupled components, that would benefit from the hybrid approach. While the business could benefit from a fast time to market granted by the hybrid approach, the author believes that the potential issues associated with complex codebase, old and new code integration, degraded performance, and larger bundle size make it unsuitable option for this migration.

After analysing common migration strategies, the author decided to proceed with the separate applications migration approach. The lack of heavily coupled components and long business processes in the project management application makes this approach a great option for this migration. This strategy allows the author to rewrite the application in a more simplistic way due to the lack of extensive integration between the old and the new code. On top of that, a smaller build size compared to the hybrid approach would benefit the users and the faster build times would make the development process more seamless.

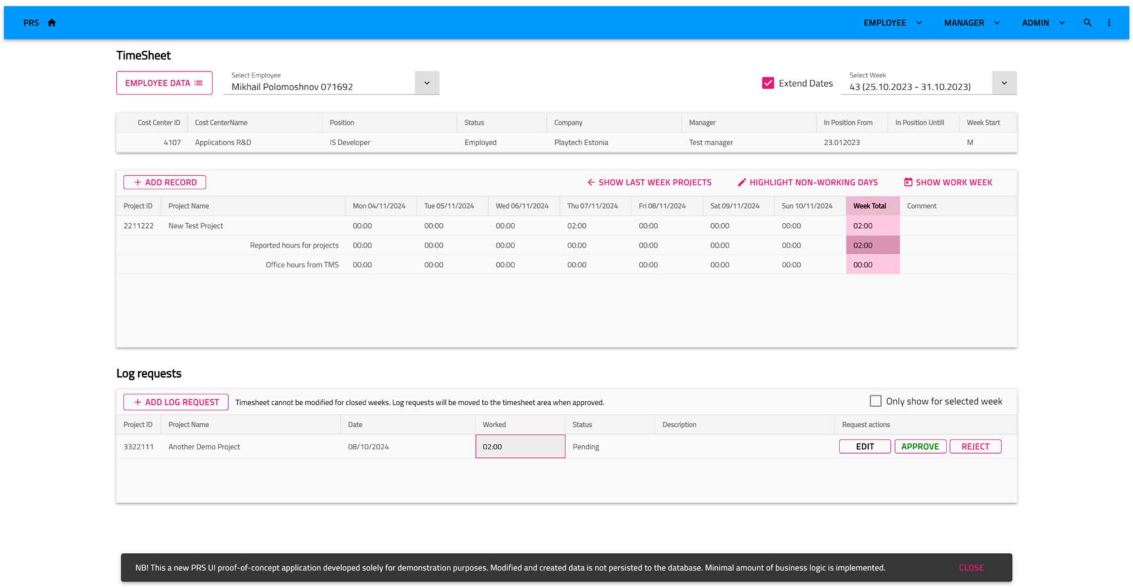
3.4 Prototype

Prototyping is the process of creating a preliminary version or a draft of an application. The prototype helps initiate the discussion and facilitate effective collaboration between different parties in the development process. It helps the parties visualise the final solution and better understand its requirements. The prototyping allows the business to save costs and time as it helps validate the ideas and fix the design flaws early on. To summarise, the prototyping reduces the chances of miscommunication, unnecessary costs and project failure. [29]

Before committing to the implementation of the new project reporting application, the author has been requested to develop a prototype by the stakeholders. The stakeholders had a blurred vision of the user interface for the new application and expressed a desire to see a visual representation of the possible solution to better understand how it might look and feel.

Before implementing the prototype, the author collected the requirements for the prototype and defined its scope. The prototype must be implemented using the default UI library for internal Angular applications at Playtech – the Ignite UI from Infragistics. The prototype must use the default styling and design language of the UI library (fluent design). The prototype must implement three commonly used pages from the old application: projects data grid, project request form and timesheet. The prototype must imitate the existing application content and layout. The prototype must use mock data and does not need to communicate with a server-side application. Finally, the business logic behind the UI elements does not need to be implemented.

The implementation of the prototype went without any complications. The author followed the requirements and developed a prototype using Angular 17 and the Ignite UI library. The implementation of the prototype took a few workdays. The prototype was deployed to a test environment and made accessible to the stakeholders (see Figure 4).



TimeSheet

EMPLOYEE DATA

Select Employee: Mikhail Polomoshnov 071692

Extend Dates

Select Week: 43 (25.10.2023 - 31.10.2023)

Cost Center ID	Cost CenterName	Position	Status	Company	Manager	In Position From	In Position Untill	Week Start
4107	Applications R&D	IS Developer	Employed	Playtech Estonia	Test manager	23.012023		M

Project ID	Project Name	Mon 04/11/2024	Tue 05/11/2024	Wed 06/11/2024	Thu 07/11/2024	Fri 08/11/2024	Sat 09/11/2024	Sun 10/11/2024	Week Total	Comment
2211222	New Test Project	00:00	00:00	00:00	02:00	00:00	00:00	00:00	02:00	
	Reported hours for projects	00:00	00:00	00:00	00:00	00:00	00:00	00:00	02:00	
	Office hours from TMS	00:00	00:00	00:00	00:00	00:00	00:00	00:00	00:00	

Log requests

Timesheet cannot be modified for closed weeks. Log requests will be moved to the timesheet area when approved.

Only show for selected week

Project ID	Project Name	Date	Worked	Status	Description	Request actions
3322111	Another Demo Project	08/10/2024	02:00	Pending		EDIT APPROVE REJECT

NB! This a new PRS UI proof-of-concept application developed solely for demonstration purposes. Modified and created data is not persisted to the database. Minimal amount of business logic is implemented.

Figure 4. Project reporting system prototype. Timesheet view.

The prototype was well received by the stakeholders and helped them better understand the requirements for the final solution. In particular, the stakeholders decided not to proceed with the default material design language. The stakeholders preferred to opt for the fluent design language instead. They understood what colour scheme should be implemented in the final solution. Moreover, the prototype made the stakeholders realise the need to engage a UI/UX professional in the view design process which will in developing a more consistent and user-friendly application. Lastly, the prototyping allowed the author to get more acquainted with the UI library and its features, which will help in the future development of the new application.

3.5 Vision of the final solution

In this section, the author will lay out the functional and non-functional requirements for the new application as the last step before the development.

The functional requirements for the new solution are dictated by the needs of the business and defined in collaboration between the stakeholders and the project manager. These two parties have the most comprehensive understanding of the business processes behind the project reporting system. The functional requirements can also be altered by input from the other parties in the process, such as the author and the author's team lead. Using their technical expertise and experience with the old system they can contribute to optimising the business logic behind the new application.

The non-functional requirements for the new solution are defined by the author in collaboration with other parties in the development process. The non-functional requirements are determined by a number of factors, namely the standards in the industry, the standards within the organisation, the pitfalls of the old solution, as well as the needs of users. The non-functional requirements for the final solution are largely based on the results of the analysis conducted by the author and the lessons learned from the prototyping.

3.5.1 Functional requirements

The high-level functional requirements for the final solution include:

- All of the regular employee views must be included in the scope of the first production deployment and must be removed from the old application
- The solution must implement the project request page, where the users can submit a project creation request to the budgeting team.
- The solution must implement the timesheet page, where the users can examine and log the time they or their subordinates spent working on a specific project.
- The solution must implement the estimations page where the users can submit time and outsourcing cost estimations for a specific project.
- The solution must implement the available projects page where the users can examine their data and the projects available to them.
- The regular employee views must also be accessible to other user groups
- The regular employee views must include a number of features only accessible to the members of the more privileged user groups.
- The solution must not break or alter the existing business processes unless it is explicitly defined in the requirements for a specific functionality.
- The functionality of each view must closely follow the low-level requirements defined by the stakeholders and the project manager.

- The managers, the budgeting and the admin employees must be able to seamlessly navigate from the new solution to the old one and vice-versa.

3.5.2 Non-functional requirements

The non-functional requirements for the final solution include:

- Application type: the application must be developed using Angular and TypeScript; the application must follow the SPA design pattern.
- Migration approach: the solution must be implemented using the incremental two-application migration approach
- Performance: the application's initial load time typically must not exceed 3 seconds in regular conditions; the user interactions must happen instantaneously or, in case of communication with the server application, must happen within 1 second; the solution must avoid unnecessary repeated requests to the server application.
- UI: the UI must implement the fluent design language; the UI must closely follow the mock-ups created by the UI designer; the UI must follow the responsive design principles; the UI must be implemented using the business unit standard Angular UI library – Ignite UI; the user interactions with the UI must provide appropriate feedback.
- Reliability: the solution must be able to provide appropriate feedback to the user in the event of partial failure of the system.
- Validation: the solution must validate the user input and provide appropriate feedback in case of incorrect input.
- Browser support: the solution must support Google Chrome; support for other browsers is optional.
- Security: the solution must prohibit anonymous authentication; the solution must implement the same authentication approach as the old application; different parts of the solution must be secured with authorization; the solution must be accessible only within the corporate network; the solution must be secured with HTTPS; the solution must pass the internal security review before the initial deployment.
- Maintainability: the solution must be implemented using the standard practices in the industry and organisation; the solution must be modular and scalable.

- Testing: the core functionalities of the new solution must be covered with unit tests
- Cost: the solution must not introduce any additional costs apart from the UI library licensing.
- Internationalisation: the final solution must only support the English language; the timesheet view must support both Monday and Sunday as the first day of the week.
- Deployment: the new solution must have the same hosting infrastructure as the old application (i.e. IIS web server running on a Windows Server VM hosted on an on-premises server).
- Compatibility: the solution must work together with the old application; the navigation from one solution to another must be seamless.

4 Implementation

In this chapter of the work, the author will introduce the work process organisation. Then, the author will provide a detailed description of the tools, technologies and architectural decisions selected for the development of the new project reporting application. Next, the author will describe the development, testing and deployment processes for the new solution.

4.1 Development process organisation

The author’s business unit follows an Agile methodology called Kanban to organise the software development process. Kanban is characterised by continuous development and delivery of features, meaning that the software development process is not split into specific time periods or cycles. To organise work in progress, a Kanban board is utilised, where ongoing tasks are split into groups by progress status. Kanban limits the amount of work in progress to maintain a steady flow of deliveries and ensure that the development team operates with respect to its capacity. Kanban does not have a strict framework to follow when it comes to meetings. However, progress update meetings, demo meetings and retrospective meetings are a regular occurrence in Kanban. [30]

Next, the author will provide a step-by-step description of the development process for the new application. First, the project manager and the stakeholders conduct a series of meetings to discuss and define the functional requirements for specific functionality. The project manager documents the requirements in project management software called Jira. The requirements are grouped into individual tasks called Jira issues, which are then added to the Kanban board.

The project manager, UI designer and stakeholders participate in a design session, where the parties communicate appearance and behaviour of a specific part of the application. The UI designer then compiles a detailed mock-up (or a set of mock-ups) in a specialised design tool called Figma (see Appendix 2 for an example).

The requirements are communicated to the developer by the project manager using Jira issues, Figma mock-ups, process-flow diagrams (see Figure 5), emails, chat messages and voice meetings. The author can provide feedback on the requirements and the mock-up designs to the project manager. The requirements and the mock-up designs are updated if needed.

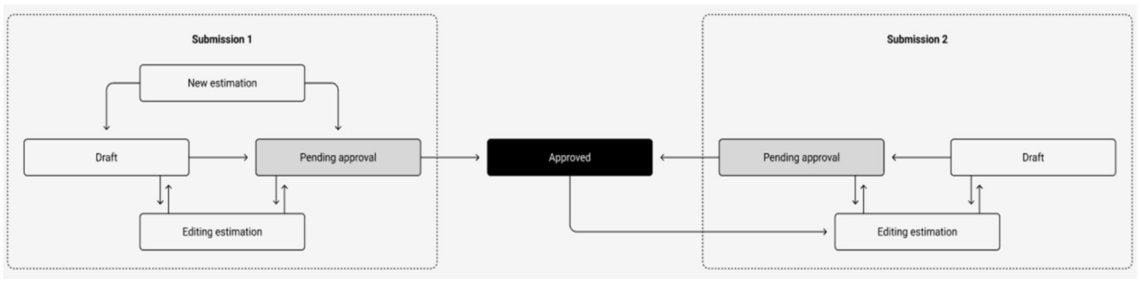


Figure 5. Project estimation process diagram.

The author develops the functionality. During the development process, regular check-up meetings are conducted to communicate the progress and clarify the requirements. As the development progresses, the author moves Jira issues to the appropriate group on the Kanban board to ensure task progress visibility.

After the specific development is finished, the author deploys it to a test environment. The solution is then extensively tested by the author and the project manager. The issues discovered during testing are fixed, and the solution is retested. Then, a demo session is organised, where the project manager presents the solution to the stakeholders. Next, the stakeholders can conduct independent testing of the solution. After testing, the stakeholders provide feedback to the project manager. The project manager communicates the feedback to the author. The author updates the solution if any adjustments are needed. The solution is tested again by all parties. The last several steps can be repeated until no more adjustments are needed. Then, the entire process is repeated for the next functionality.

4.2 Development

In this section, the author will describe the development process for the new application. In particular, utilised tooling, architectural decisions, new application development, and old application compatibility changes will be covered.

4.2.1 Development tooling

The author utilised a good number of tools to facilitate the development of the new application. The tools selected by the author are considered to be standard in Angular development. The author tried to minimise custom configuration to the tooling to enhance the maintainability and consistency of the codebase.

The author used a code editor called Visual Studio code to write the code. The author has installed several extensions to the code editor. The ESLint extension is a popular code linting utility which statically analyses the code for common programming and stylistic errors. The Prettier is a commonly used code formatter that ensures that code is styled consistently, making it easier to read and maintain. The author used Angular Language Service extensions for the framework-specific code suggestions and templates. The author used the Ignite UI Tooltips and Toolbox extensions for the UI library-specific code suggestions and templates. The author used an AI-powered GitHub Copilot extension for code suggestions and automation of some routine programming tasks (e.g. regex generation, debugging).

The author utilised Angular CLI – a command-line interface tool that allows developers to scaffold, build, test, deploy, and maintain Angular applications from a command shell. For source code version control and hosting the author used Git in tandem with Gitlab.

4.2.2 Architectural and technological decisions

In this subsection, the author will provide a brief overview of the technologies and architectural approaches behind the new application. As a rule, the author will try to adhere to the industry standards to enhance code readability and maintainability. The cases when the author opts for more complex or non-default approaches will be covered in more detail in this section. Apart from the standards, the technological and architectural decisions are dictated by requirements for the application and results of the analysis.

For the new application, the author will use Angular version 18 – the latest Angular version available at the development start date. The code for the application will be written in TypeScript – a typed superset of JavaScript. During the development, the author will often use Angular components – the base building blocks of Angular applications. Components define what is rendered on a page or a part of a page, as well as how the rendered content looks and behaves. The component consists of an HTML template that defines component content, a TypeScript class that defines component behaviour, and optional styling. By convention, all parts of the component are defined in separate files.

The application will use custom UI controls, components and styling options provided by the Ignite UI library to facilitate the development of the new application. As previously stated, Ignite UI is the Angular UI library of choice in the author's business unit. Using the same library for different projects helps provide consistent UI across different applications and eases the maintainability of the applications. The library uses SASS by default – a superset of CSS that provides additional features to help better organise styling in the application. The application will also adopt SASS for styling purposes.

With the Angular 17 release, the Angular team introduced a new architectural approach for developing applications – standalone components. With the new approach, the application no longer depends on the Angular modules – "container" files that encapsulate and manage dependencies for related components, directives and services. The standalone components approach provides a simpler and more descriptive way to build Angular applications. It allows components to manage their dependencies, eliminating the need for Angular modules. This approach simplifies lazy loading, reduces the amount of boilerplate code, and decreases the bundle size and build time [31]. Furthermore, this approach will become the default way to build Angular applications with the Angular 19 release. For all these reasons, the author will use the standalone components approach for the project reporting application.

The author will utilise the default Angular router to organise routing in the application. The router allows the developer to define what parts of an application are rendered when the user navigates to a particular URL. The default router provides the developer with a rich set of features, including support for authorization, state management and lazy loading.

The author will utilise the service pattern to encapsulate often used code or code specialised on one specific task to promote code reusability and separation of concerns. Angular service is a TypeScript class that includes a special decorator to help the Angular dependency injection service recognise the class as an injectable service. The Angular DI service manages the service lifecycle and facilitates the interaction with the dependency consumer. The DI service helps write more loosely coupled code by implementing the inversion of control principle.

The data services in the new application will use Angular HTTP service, which provides the base functionality needed for composing and executing HTTP requests. The data services will be derived from the base service. The base service will implement functionality shared by the data services. This approach promotes code reusability and provides a consistent way to handle common data request tasks (e.g. error handling).

To enhance the user experience and optimise the overall performance of the system, the author will utilise the NgRX library for state management. NgRX follows the Redux pattern – a well-established approach for global state management in JavaScript applications. NgRX provides developers with a standardised and performant way to store the data shared across the application. NgRx offers a wide range of features, including side-effects handling, global store, router support, custom operators, and entity store adaptors. While the NgRx library is complex and requires more boilerplate code than other state management libraries or custom solutions, its rich feature set, well-established standardised approach and popularity made it the ultimate choice for state management in the new project reporting application.

For asynchronous programming scenarios, such as HTTP requests, state management and UI interaction handling, the application will utilise a library called RxJS. The library offers a broad set of features for asynchronous development and is a de facto standard in Angular development. Another officially supported library for asynchronous programming in Angular is called Angular Signals. Angular Signals is a new library that provides a more simplistic and intuitive way of writing asynchronous code compared to RxJS. However, it is still not supported by all the core Angular libraries and is not as feature rich as RxJS. Therefore, the author decided to persist with the RxJS library for this development.

Angular offers two different approaches for creating forms: template-driven forms and reactive forms. The template-driven forms are best suited for simple forms and two-way data binding scenarios. The reactive forms approach is suitable for more sophisticated forms that include dynamic form control creation, control grouping and complex validation. The author will utilise both approaches in the project reporting application depending on the requirements for a specific form.

For unit testing in the application, the author will use the default testing tools for Angular: Jasmine test framework and Karma test runner.

4.2.3 Application base development

The author started the development by using Angular CLI to scaffold the base of the application and install Angular dependencies. The CLI generated the basic folder structure, configuration files, entry point files and root component of the application. The author then installed and set up other relevant dependencies, such as the state management library and the UI library.

The author added the text and icon font imports to the global stylesheet of the application. The author added the Playtech logo that will be used in the navigation bar to the static assets folder. Using the custom SASS mixins and functions provided by the UI library, the author configured the global styling for the application, including the colour scheme, typography and design language (see Appendix 3).

Using the Angular CLI, the author scaffolded a component for each regular employee view that will be implemented. The author also scaffolded components for the navigation bar and error views (e.g. page not found, unauthorized, forbidden).

The author started configuring the routing for the application. The author added routes for the regular employee views and the error views. The author set up parametrised routes, the default route that navigates the user to the timesheet page, and the wildcard route that navigates the user to the “page not found” view (see Appendix 4). The author added the router and the routes to the application configuration. Lastly, the author added the router outlet to the application root component.

The author added content to the error and navigation bar components. The error views include simple error messages and will be rendered in case the requested page is not found as well as if the authentication or authorization failed. The navigation bar contains the Playtech logo, employee views dropdown menu and welcome message (see Figure 6). For the admin, budgeting and manager users, the navigation bar also displays a button that will navigate them to the old application. The author added the navigation bar to the application root component to ensure that it is always rendered on the page.

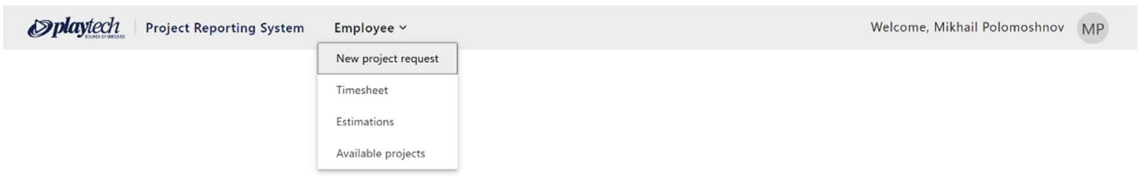


Figure 6. Application navigation bar.

The author started setting up the data services by implementing the base service. Then, the author added data models and implemented two data services that will be used to retrieve the global state. The author added the environment files that will store the environment-specific constants, including the back-end API base URLs. The author created a file that will store the API endpoint routes so that they can be managed from one place. The author added an HTTP request interceptor, which includes the session cookie in cross-site HTTP requests. This step is crucial, as the new front-end and server-side applications will be hosted on different sites. Lastly, the author added the HTTP client and interceptor to the application configuration.

The author started setting up the application state. The author created an interface, reducer, actions, selectors, and side effects for the shared state, which will hold the data used in most of the components across the application. The shared state includes the basic user data and the list of employees. The state side effects utilise the data services developed earlier. The author added the shared state and side effects to the application configuration file to ensure that the shared state will be initialised on the initial application load.

As the next step, the author set up authorization and authentication. For this purpose, the author developed a special route guard (see Appendix 5). The guard uses the user data from the shared state and the “allowed” role provided in the route configuration to determine whether the user can access a specific route. If the user is not authenticated or unauthorized to access a view, the guard navigates the user to an appropriate error page. After implementing the guard, the author secured the application routes with it (see Appendix 4).

The author updated the project's Angular configuration. In particular, the author configured the build settings for different environments and the base URL for the that will be used in all relative paths of the application. This step is crucial as both the old and the new applications will be served on the same site. The new application will be available under the "v2" path.

Lastly, the author implemented unit tests that cover the core functionalities of the application.

4.2.4 Application views development

On a high level, the development of each view follows a similar pattern. The author develops a specific part of the view's component by implementing its template, code and styling. The author imports the relevant framework and library dependencies to the component. The author develops the data models and services that are required for the specific part of the component. The author sets up the state by creating or updating the interface, reducer, actions, selectors and side effects of the component-specific slice of the state. The author updates the routing setup if required. The steps are repeated until all parts of the view are implemented. Lastly, the author implements unit tests for the core functionalities of the component and the error handling. The error handling for each view is implemented in the same way. In case of server-side errors or user errors (e.g. invalid route parameters) the pop-up notification with an appropriate error message is displayed to the user. Next, the author will describe the functionality and distinctive details of each implemented view.

4.2.5 New project request view

The new project request view provides the user with a form for submitting a project request (see Appendix 6). The form consists of different form controls, including the input fields, text boxes, combo boxes, checkboxes, drop-down lists, hierarchical multi-select drop-down lists and radio buttons.

From the technical perspective, the main difference between this and the other views is that the project request form is implemented using the reactive forms approach. For this view, the author implemented inline field validation as well as field group validation. The author also implemented dynamic field group rendering based on the selected project group value. The author implemented dynamic value generation for the Project Name

Group and Project Full Name fields based on the values provided in five different fields. The view has a small amount of custom styling and supports field column wrapping to better accommodate different screen sizes and resolutions.

From the functional point of view, the new page is similar to the same page in the old application. The new view implements better validation and a more compact interface for selecting required Jira accounts. The old server-side validation has been replaced with the client-side inline validation, enhancing the user experience. Moreover, the validation in the new view is more thorough and covers all the required validation scenarios.

Some of the budgeting and administrator views developed in the future will share functionality with the project request view. For this reason, the project request view has been developed with the code reusability in mind. In particular, the dynamic field value generation logic and the field group validation logic were implemented as separate services. Additionally, the project state slice created during this development will be reused in other project views in the future.

4.2.6 Timesheet view

The timesheet view allows users to examine and log the time they or their subordinates spent working on a project. The Timesheet view also provides functionality to add work log comments and examine user data.

The view has two distinct modes – the week mode and the month mode. The month mode presents the data in calendar format, where the user can get a quick overview of the time reported in a specific month (see Appendix 7). The user can select a date to see and edit individual logs for that day. The week mode presents the data in a hierarchical tree grid format displaying the work log data for a specified week (see Appendix 8). The tree grid allows the user to edit the work logs and work log comments. The tree grid also displays different summaries of hours reported for the week.

The view implements different work log display and edit options depending on the work log date and source. The view supports different week starting days depending on the user’s country. The view exposes different data read and edit options for different user groups. The days in both modes are colour-coded. Different colours represent different day types, such as sick leave, vacation, holiday, weekend and regular days.

The main distinguishing feature of this view compared to other implemented views is that it required a lot of styling and extensive customisations of the default behaviour of the components provided by the UI library. In particular, the author developed custom rendering logic for the Ignite UI calendar day view component to implement the timesheet month mode.

From the functional point of view, the new timesheet is very different from the same page in the old application. First, the business process behind time reporting has changed, which resulted in the log request grid being removed from the view. Secondly, the brand-new month mode has been introduced. Lastly, the week mode has been heavily updated, reducing the number of pop-up windows required to input the data.

4.2.7 Estimations view

The estimations view provides the user with the functionality to submit time and outsourcing cost estimations for projects (see Appendix 9). The page is represented by a simple form that consists of drop-down lists, combo boxes, text boxes, and input fields. The form supports custom drop-down value filtering and simple validation. The form has been implemented using the template-driven approach.

From the functional perspective, the new view is slightly different from the old estimation view. In particular, the business process behind the outsourcing cost estimation has changed which resulted in a new section being added to the form.

4.2.8 Available projects view

The available project view allows the user to examine the available project details and personal data of employees (see Appendix 10). The employee data is displayed in a simple table, while the project data is displayed in a master-detail style grid. The view exposes different data to the user based on the user's role. The available project grid supports dynamic column addition, hiding, sorting and filtering.

From the functional perspective, the main difference between the old and new views is that the new view exposes more information about the projects in the detail section of the grid. The old solution implemented a simple grid with a minimal amount of project data. Additionally, the new solution implemented column hiding and more sophisticated options for column filtering and sorting. Lastly, the new solution allows the user to load an extended project list on demand.

4.2.9 Old application compatibility changes

The author updated the old application to ensure smooth interoperability between the old and new solutions. The author removed the migrated views from the old application and added a button to its navigation bar that redirects users to the new application. Since the old and new applications are hosted on the same site, the author configured the IIS URL rewriting. The new application will be available under the "v2" route. The author set up the new timesheet view as the landing page. The author set up redirection to the new application for the links removed from the old application. This way, the old links will remain functional, and the users will be redirected to the corresponding pages in the new application. The URL rewrite configuration created by the author also supports parametrised routes. The links for the remaining views in the old application will remain functional.

4.3 Testing and validation

Before deploying the final solution to the production environment, thorough testing must be conducted. As described in the work process organisation section of this chapter, the parties of the development process conducted testing after each stage of the development. When the development of the regular employee views had been finalized, the solution was deployed to the test environment for the final round of testing.

The final round of testing was conducted by different parties, including the author, project manager, stakeholders, and a selected group of other application users. The testers represented all the existing user groups. The testing covered both the default usage scenarios and edge cases. The testing focused on error handling, validation, interoperability with the old application, as well as the user role and country-related customisations. During the testing process, the parties also reviewed the non-functional aspect of the new application. In particular, the testers examined the application's appearance, security, performance, user interaction feedback, and usability with different screen sizes.

During the testing process, a few bugs were discovered, and several enhancements were proposed by the testers. The author fixed the bugs and implemented the relevant enhancements. Then, the solution was retested. The testing parties verified that the solution works according to the specified requirements and that all application features perform correctly. Consequently, the author received the deployment approval from the stakeholders.

Apart from the stakeholder approval the application also needed to pass the review from the internal security team to ensure that the application complies with the company security standards. The author and his team lead prepared an overview of the upcoming application migration and submitted it for a security evaluation. The internal security team reviewed the upcoming transition and granted their approval for the deployment.

4.4 Deployment

The deployment process for the new front-end application was uncomplicated. Before the deployment of the new application, the author put together a rollback plan.

The author prepared production builds for the new and old applications. The author deployed the build files of the applications and the new URL rewrite configuration to the production environment using the IIS Web Deploy tool.

After the deployment, the author performed some tests in production environment and participated in post-deployment monitoring of the new application. The deployment was successful, and the author did not encounter any complications.

5 Results

As a result of this work, the main objectives described in section 2.6 have been achieved – the new project reporting front-end application has been developed and deployed to the production environment. The new application provides a comprehensive solution to the problems described in section 2.5 and satisfies the requirements outlined in the section 3.5. The solution implements the most used views of the old application and lays the foundation for future migration efforts.

The new application demonstrates substantial improvements over its predecessor in several key areas. With regard to performance, the new solution operates more efficiently and swiftly. The user interface has been thoroughly redesigned offering the users more modern, intuitive and visually appealing experience. The user experience is further enhanced by improved responsiveness and feedback provided by the new solution. The new application is designed to address the current needs of the organisation and implements refined and enhanced business logic.

The new solution is implemented using state-of-the-art front-end technologies, improving application security, efficiency and reliability. The application codebase maintainability and scalability are enhanced by using modern technologies, best development practices and established industry standards. All these factors ensure that the new solution is well-positioned for continued development and future growth.

The users who participated in testing or utilised the new application post-deployment have consistently provided positive feedback, highlighting the improvements listed in the second paragraph of this chapter. Following the deployment, the author will continue collecting and analysing feedback from users to further improve the new solution.

5.1 Efficiency comparison

This section of the work will present a performance comparison between the old and new solutions, focusing on key metrics relevant to front-end applications. First, the application bundle total transfer size has reduced from 5.25 MB to 0.7 MB. This includes the application code, markup, styling and static assets. Though the new solution does not yet implement all of the functionalities from the old application, it will employ lazy loading for all the views that will be developed in the future. This will ensure that most users continue benefiting from the drastically reduced bundle size.

Another critical area where the new application demonstrates efficiency improvements is the number of repeated requests to the server-side application. Unlike the old application, the new solution implements a global store that can act as a cache, holding data that has been previously fetched. The new solution eliminates the need for redundant queries to the server for the same information and decreases the view content load time. This, in turn, enhances the user experience and overall performance of the system. The following table demonstrates the reduction in the number of reoccurring requests when navigating to different views in the new application.

Table 2. New application. Initial and repeated requests.

Scenario	**New project request**	**Timesheet**	**Estimations**	**Available projects**
Number of requests when first navigating to the view	8	3	2	2
Number of requests when navigating back to the view	1	3	0	0

Lastly, the author tested and compared the performance of the new and old solutions using Google Lighthouse – a front-end application auditing tool. During testing, the author focused on the following key metrics evaluated by the tool: first contentful path, speed index, JavaScript execution time and total blocking time. The first contentful path metric represents the time it takes for the first piece of content to render on the screen. The speed index shows how quickly all of the page contents are populated. The JavaScript execution time represents the time the browser spends executing JavaScript to load the page. Lastly, the total blocking time measures the time the page is blocked from responding to user input during loading. The second metric is heavily dependent on the server-side application performance, and best demonstrates the overall efficiency of the system. The other metrics mostly focus on the front-end application performance.

Google Lighthouse uses simulated browser context for performance evaluation. The context resource and network configuration are not optimal by design, meaning that the auditing tool tries to simulate page loading on a less performant device with a slower network speed. Therefore, the test result must not be used to evaluate the performance experienced by an average user of the applications but rather to compare the efficiency of two solutions in a controlled environment. [32]

The tests were conducted on the same machine and network in a one-hour session. Both applications are hosted on the same test server that was not being actively used during the testing. The tests were carried out in the Google Chrome browser, build 131.0.6778.86. The author ran the test in incognito mode to ensure that the results were not affected by the browser plugins.

The test results can be found in appendices 11 and 12 (see Table 3 and Table 4). The metrics demonstrated in the tables represent the average result from 10 test runs. The new application demonstrates significantly better performance across all key metrics. The new solution is more responsive, consumes less resources and loads faster, providing a superior user experience.

5.2 Challenges and complications encountered

The main challenges encountered during the new project reporting application development are related to the business processes behind the system.

Some of the old application business logic has not been sufficiently documented and was lost due to employee turnover. During the development process, the author often had to analyse the legacy code, and the project manager had to investigate old Jira issues in search of the business rules that might be relevant to the new application. This, in turn, slowed down the delivery process.

Another challenge encountered by the author is related to the changing requirements for the new application. At times, new requirements were introduced during the testing stage. While the new additions helped enhance the application, they also increased the scope of the development, pushing the initial deployment date further.

5.3 Future direction

The logical progression of the project reporting system development involves continuous migration of the views from the old front-end application to the new one. The first probable candidate for migration is the manager user group functionality.

One area the project reporting application could improve on in the future is the automated test coverage. Due to the business constraints and a tight deadline, only the core functionalities of the application have been covered with tests. Enhancing this aspect will be crucial for improving the robustness, maintainability and reliability of the application in the long-term perspective.

Another logical improvement to the new application could be the implementation of continuous integration and continuous deployment. This enhancement will automate testing, build and deployment of the new application enhancing the efficiency and reliability of the delivery processes.

The author should remain informed on the latest advancements and trends in the Angular ecosystem and incorporate new developments into the application. Finally, the author should regularly update the application's Angular version to enhance its longevity, performance and maintainability.

6 Summary

In the scope of this work, the author conducted an in-depth examination of the migration process for a legacy front-end application from various perspectives. The main objective of this work was to formulate a migration approach and migrate the legacy application.

In the analysis part, the author examined the old application and compared common technological solutions and migration strategies from various aspects. As a result, Angular was selected as the main framework for the new solution, and an incremental rewrite with two separate applications was chosen as the migration approach. Then, the author developed and validated a prototype for the final solution. Subsequently, the author analysed functional and non-functional requirements for the new application. In the analysis, the author gave thorough consideration to the needs of the organisation and the specific requirements of the project reporting system.

The implementation part of the thesis predominantly focused on the process of solving the problem of the work. In this part, work process organisation, utilised tooling, low-level technological decisions, and the process of development, testing and deployment were covered.

As a result of this work, the new solution was developed and deployed to the production environment, fulfilling the main objective of the thesis. The new solution is implemented using state-of-the-art front-end technologies and standard programming approaches, providing a comprehensive solution to the main problem of this work. The new solution demonstrates major improvements over the legacy application across all key areas. The new application is more performant, secure, and visually appealing, providing a superior experience to the users. The new application implements refined and enhanced business logic. The new application is more maintainable and is well-positioned for continued development and future growth.

References

[1] Microsoft, “What is a browser-based application?,” Microsoft, 11 April 2023. [Online]. Available: https://www.microsoft.com/en-us/edge/learning-center/what-is-a-browser-based-application?form=MA13I2. [Accessed 28 September 2024].

[2] Amazon Web Services, “What’s the Difference Between Frontend and Backend in Application Development?,” Amazon Web Services, 2024. [Online]. Available: https://aws.amazon.com/compare/the-difference-between-frontend-and-backend/. [Accessed 28 September 2024].

[3] Mozilla Corporation, “What is JavaScript?,” Mozilla Corporation, 3 August 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Accessed 2024 September 28].

[4] R. Vyas, “Comparative Analysis on Front-End Frameworks,” *International Journal for Research in Applied Science & Engineering Technology,* vol. 10, no. 7, pp. 298-307, 2022.

[5] Playtech, “About Us - Playtech,” Playtech, 2024. [Online]. Available: https://www.playtech.com/about-us/. [Accessed 20 September 2024].

[6] Google, “AngularJS: Miscellaneous: Version Support Status,” Google, 2020. [Online]. Available: https://docs.angularjs.org/misc/version-support-status. [Accessed 20 September 2024].

[7] XLTS.dev, “Extended Long-term Support for AngularJS,” HeroDevs, 2024. [Online]. Available: https://xlts.dev/angularjs. [Accessed 28 September 2024].

[8] B. Świątek, Ł. Guzek and A. Rosłoniec, “AngularJS upgrade: Why 2023 is the time for migrating and how to do it,” Pretius, 16 November 2023. [Online]. Available: https://pretius.com/blog/angularjs-upgrade/. [Accessed 28 September 2024].

[9] G. Lipke, “Angular vs AngularJS — Why you should migrate from AngularJS to Angular,” House of Angular, April 2024. [Online]. Available: https://houseofangular.io/angular-vs-angularjs-why-you-should-migrate-from-angularjs-to-angular/. [Accessed 28 September 2024].

[10] S. S. B. Praneeth, “Survey on migrating Angular JS applications to Angular 2+ versions,” *International Journal of Advances in Engineering and Management (IJAEM),* vol. 4, no. 7 July 2022, pp. 1615-1618, 2022.

[11] OpenJS Foundation, “jQuery,” OpenJS Foundation, 2024. [Online]. Available: https://jquery.com/. [Accessed 28 Septemberr 2024].

[12] Google, “AngularJS: Developer Guide,” Google, 2020. [Online]. Available: https://docs.angularjs.org/guide. [Accessed 28 September 2024].

[13] N. Meurer, “A Primer for $http in AngularJS,” Natalie Meurer, 14 December 2014. [Online]. Available: https://natmeurer.com/put-logic-in-your-factories-a-primer-for-http-in-angularjs/. [Accessed 28 September 2024].

[14] Angular University, “Angular Service Layers: Redux, RxJs and Ngrx Store - When to Use a Store And Why?,” Angular University, 17 January 2024. [Online]. Available: https://blog.angular-university.io/angular-2-redux-ngrx-rxjs/. [Accessed 28 September 2024].

[15] OpenLogic, “AngularJS Long-Term Support,” OpenLogic, 2024. [Online]. Available: https://www.openlogic.com/solutions/angularjs-support-and-services. [Accessed 28 September 2024].

[16] W. Xu, “Benchmark Comparison of JavaScript Frameworks,” University of Dublin, Dublin, 2021.

[17] N. Shawn, “Unveiling the Best Frontend Framework for You in 2024: A Deep Dive,” 2 May 2024. [Online]. Available: https://www.linkedin.com/pulse/unveiling-best-frontend-framework-you-2024-deep-dive-naziullah-shawn-agzlc/. [Accessed 28 September 2024].

[18] T. Krotoff, “Front-end frameworks popularity (React, Vue, Angular and Svelte),” 12 Decemberr 2023. [Online]. Available: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190#file-frontendframeworkspopularity-md. [Accessed 2024 September 28].

[19] State of JavaScript, “State of JavaScript: Front-end Frameworks,” State of JavaScript, 2023. [Online]. Available: https://2023.stateofjs.com/en-US/libraries/front-end-frameworks/. [Accessed 28 September 2024].

[20] H. Harnisch and K. Womersley, Atomic Migration Strategy for Web Teams, June: O'Reilly Media, Inc, 2018.

[21] O. Gierszal, “Big Bang Migration vs Trickle Migration Approach in Legacy Modernization [Key Differences & Considerations],” Brainhub, 27 August 2024. [Online]. Available: https://brainhub.eu/library/big-bang-migration-vs-trickle-migration. [Accessed 28 September 2024].

[22] V. Savkin, Upgrading Angular Applications, May: 25, 2017.

[23] M. Jansky, “Migrating AngularJS to Angular with a hybrid application,” Codurance, 24 May 2023. [Online]. Available: https://www.codurance.com/publications/migrating-angularjs-to-angular. [Accessed 28 September 2024].

[24] Frontend Mastery, “A guide to frontend migrations,” Frontend Mastery, 17 July 2022. [Online]. Available: https://frontendmastery.com/posts/frontend-migration-guide/#migration-strategies-gradual-vs-big-bang. [Accessed 28 September 2024].

[25] Google, “Angular elements overview,” Google , 2024. [Online]. Available: https://angular.dev/guide/elements. [Accessed 28 September 2024].

[26] Google, “Upgrading from AngularJS to Angular,” Google, 2024. [Online]. Available: https://v17.angular.io/guide/upgrade. [Accessed 28 Septemberr 2024].

[27] C. B. R. P. T. Leadbetter, Cambridge International AS and A Level Computing Coursebook, Cambridge: Cambridge University Press, 2012.

[28] S. Wainwright and C. Leadbetter, Computer Studies and Information Technology: IGCSE and O Level.

[29] N. Moses, L. Wojciechowski, S. Daly and K. Sienko, “Front-end design prototyping,” *Design Science, Cambridge University Press,* vol. 9, no. 24, pp. 1-27, 2023.

[30] Atlassian, “Kanban,” Atlassian, 2024. [Online]. Available: https://www.atlassian.com/agile/kanban. [Accessed 28 October 2024].

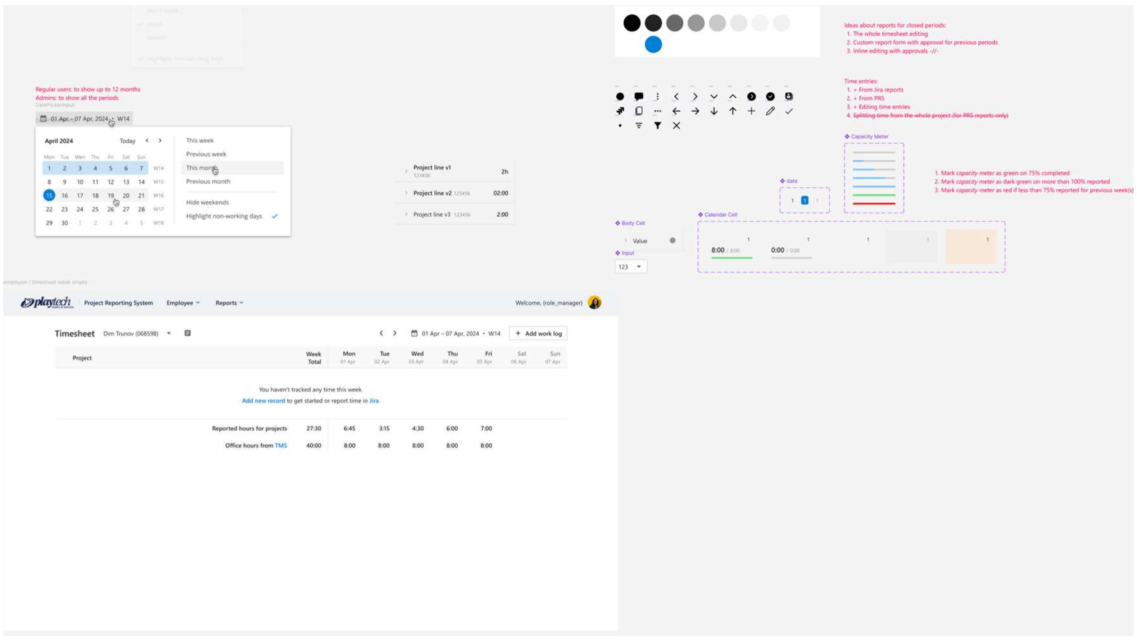
[31] Google, “Getting started with standalone components,” Google, 2024. [Online]. Available: https://v17.angular.io/guide/standalone-components. [Accessed 28 October 2024].

[32] S. Rios, “How to read a Lighthouse report and use it to optimize your loading speed,” Oncrawl, 14 May 2021. [Online]. Available: https://www.oncrawl.com/technical-seo/lighthouse-report-optimize-loading-speed/. [Accessed 28 October 2024].

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – Timesheet view Figma board



Regular users: to show up to 12 months
Admins: to show all the periods

Ideas about reports for closed periods:
1. The whole timesheet editing
2. Custom report form with approval for previous periods
3. Inline editing with approvals -//-

Time entries:
1. + From Jira reports
2. + From PRS
3. + Editing time entries
4. ~~Splitting time from the whole project (for PRS reports only)~~

1. Mark *capacity meter* as green on 75% completed
2. Mark *capacity meter* as dark green on more than 100% reported
3. Mark *capacity meter* as red if less than 75% reported for previous week(s)

Timesheet Dim Trunov (068598)

Project	Week Total	Mon 01 Apr	Tue 02 Apr	Wed 03 Apr	Thu 04 Apr	Fri 05 Apr	Sat 06 Apr	Sun 07 Apr
You haven't tracked any time this week. Add new record to get started or report time in Jira.								
Reported hours for projects	27:30	6:45	3:15	4:30	6:00	7:00		
Office hours from TMS	40:00	8:00	8:00	8:00	8:00	8:00		

Figure 7. Figma board, timesheet view mock-ups.

Appendix 3 – Global stylesheet

```
@use "@infragistics/igniteui-angular/theming" as *;

$custom-toast-theme: toast-theme(
  $background: color($light-fluent-palette, "error"),
);

$custom-snackbar-theme: snackbar-theme(
  $background: color($light-fluent-palette, "success"),
);

$my-palette: palette(
  $primary: color($light-fluent-palette, "primary", 900),
  $secondary: color($light-fluent-palette, "secondary", 900),
  $surface: #fff,
  $info: color($light-fluent-palette, "info"),
  $success: color($light-fluent-palette, "success"),
  $error: color($light-fluent-palette, "error"),
  $warn: color($light-fluent-palette, "warn"),
);

@include core();
@include typography(
  $font-family: '"Segoe UI", "Open Sans", sans-serif',
  $type-scale: $fluent-type-scale
);
@include fluent-light-theme($my-palette);
@include css-vars($custom-toast-theme);
@include css-vars($custom-snackbar-theme);

@import url("https://fonts.googleapis.com/css?family=Open+Sans");
@import url("https://fonts.googleapis.com/icon?family=Material+Icons");

body {
  margin: 0px;
}

igx-tree-grid .igx-snackbar {
  display: none !important;
}
```

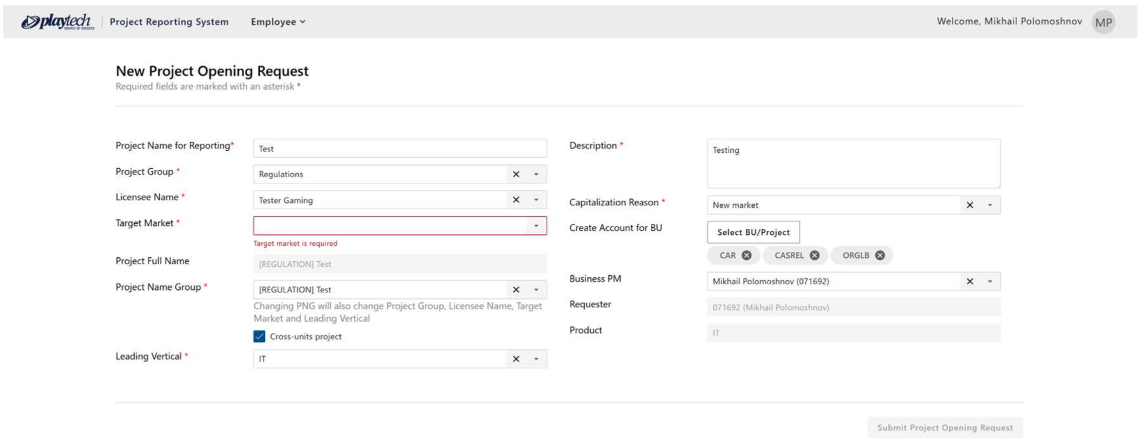
Appendix 4 – Routing configuration

```
import { Routes } from '@angular/router';
import { NewRequestComponent } from './components/projects/new-request/new-
request.component';
import { provideState } from '@ngrx/store';
import { provideEffects } from '@ngrx/effects';
import { projectReducer } from './state/project/project.reducer';
import { ProjectEffects } from './state/project/project.effects';
import { NotFoundComponent } from './components/error-pages/not-found/not-
found.component';
import { ForbiddenComponent } from './components/error-
pages/forbidden/forbidden.component';
import { InternalServerErrorComponent } from './components/error-
pages/internal-server-error/internal-server-error.component';
import { authGuard } from './shared/guards/auth.guard';
import { Role } from './model/enum/role';
import { TimesheetComponent } from
'./components/timesheet/timesheet.component';
import { timesheetReducer } from './state/timesheet/timesheet.reducer';
import { TimesheetEffects } from './state/timesheet/timesheet.effects';
import { EstimationComponent } from
'./components/estimations/estimation/estimation.component';
import { estimationReducer } from './state/estimation/estimation.reducer';
import { EstimationEffects } from './state/estimation/estimation.effects';
import { AvailableProjectEffects } from './state/available-project/available-
project.effects';
import { availableProjectReducer } from './state/available-project/available-
project.reducer';
import { AvailableProjectsComponent } from './components/projects/available-
projects/available-projects.component';

export const routes: Routes = [
  {
    path: 'timesheet',
    title: 'Timesheet',
    component: TimesheetComponent,
    canActivate: [authGuard],
    data: { role: Role.EMPLOYEE },
    providers: [
      provideState('timesheet', timesheetReducer),
      provideEffects([TimesheetEffects]),
    ],
  },
  {
    path: 'availableprojects',
    title: 'Available Projects',
    component: AvailableProjectsComponent,
    canActivate: [authGuard],
    data: { role: Role.EMPLOYEE },
    providers: [
      provideState('availableproject', availableProjectReducer),
      provideEffects([AvailableProjectEffects]),
    ],
  },
  {
    path: 'estimations',
    title: 'Estimations',
    component: EstimationComponent,
    canActivate: [authGuard],
    data: { role: Role.EMPLOYEE },
    providers: [
      provideState('estimation', estimationReducer),
      provideEffects([EstimationEffects]),
    ],
  },
  {
    path: 'projects/requests/new',
    title: 'New Project Request',
    component: NewRequestComponent,
    canActivate: [authGuard],
    data: { role: Role.EMPLOYEE },
    providers: [
      provideState('project', projectReducer),
      provideEffects([ProjectEffects]),
    ],
  },
  {
    path: 'projects/requests/new/:jiraIssueKey/:name/:jiraKey',
    title: 'New Project Request',
    component: NewRequestComponent,
    canActivate: [authGuard],
    data: { role: Role.EMPLOYEE },
    providers: [
      provideState('project', projectReducer),
      provideEffects([ProjectEffects]),
    ],
  },
  { path: '404', component: NotFoundComponent },
  { path: '403', component: ForbiddenComponent },
  { path: '500', component: InternalServerErrorComponent },
  { path: '', redirectTo: 'timesheet', pathMatch: 'full' },
  { path: '**', redirectTo: '/404', pathMatch: 'full' },
];
```

Appendix 5 – Authentication and authorization route guard

```
import { inject } from '@angular/core';
import { CanActivateFn, Router } from '@angular/router';
import { Store } from '@ngrx/store';
import { combineLatest, map } from 'rxjs';
import { selectError, selectUser } from
'../../state/shared/shared.selectors';

export const authGuard: CanActivateFn = (route) => {

  if (!route.data['role']) return false;

  const store = inject(Store);
  const router = inject(Router);
  const userState$ = store.select(selectUser);
  const userError$ = store.select(selectError);

  return combineLatest([userState$, userError$]).pipe(
    map(([user, error]) => {
      if (error || (!user.role)) {
        router.navigate(['/401']);
        return false;
      }
      if (route.data['role'] <= user.role) {
        return true;
      } else {
        router.navigate(['/403']);
        return false;
      }
    })
  );
};
```

Appendix 6 – New project request view


Project Reporting System | Employee | Welcome, Mikhail Polomoshnov

New Project Opening Request

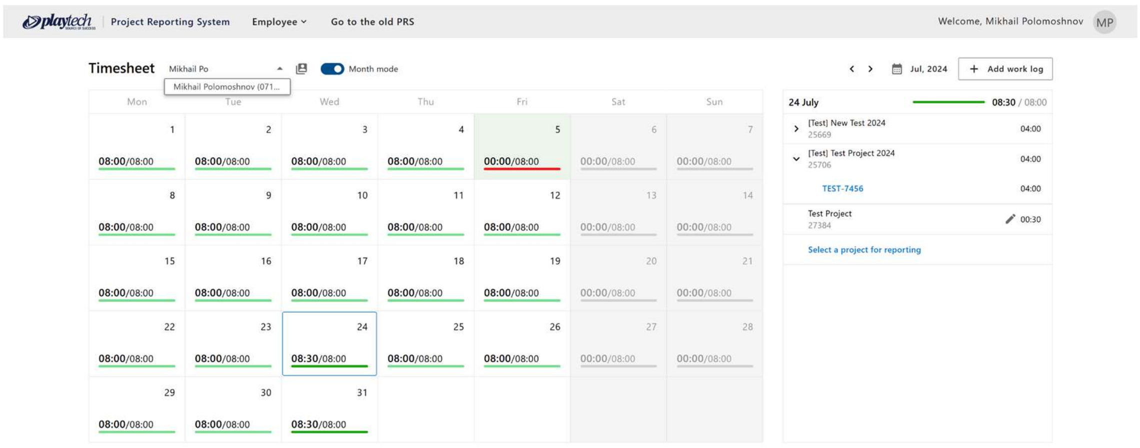
Required fields are marked with an asterisk *

Project Name for Reporting*	Test
Project Group *	Regulations
Licensee Name *	Tester Gaming
Target Market *	Target market is required
Project Full Name	[REGULATION] Test
Project Name Group *	[REGULATION] Test Changing PNG will also change Project Group, Licensee Name, Target Market and Leading Vertical ✓ Cross-units project
Leading Vertical *	IT
Description *	Testing
Capitalization Reason *	New market
Create Account for BU	Select BU/Project CAR, CASREL, ORGLB
Business PM	Mikhail Polomoshnov (071692)
Requester	071692 (Mikhail Polomoshnov)
Product	IT

Submit Project Opening Request

Figure 8. New project request view.

Appendix 7 – Timesheet view, month mode



Timesheet Mikhail Po — Mikhail Polomoshnov (071...

Month mode

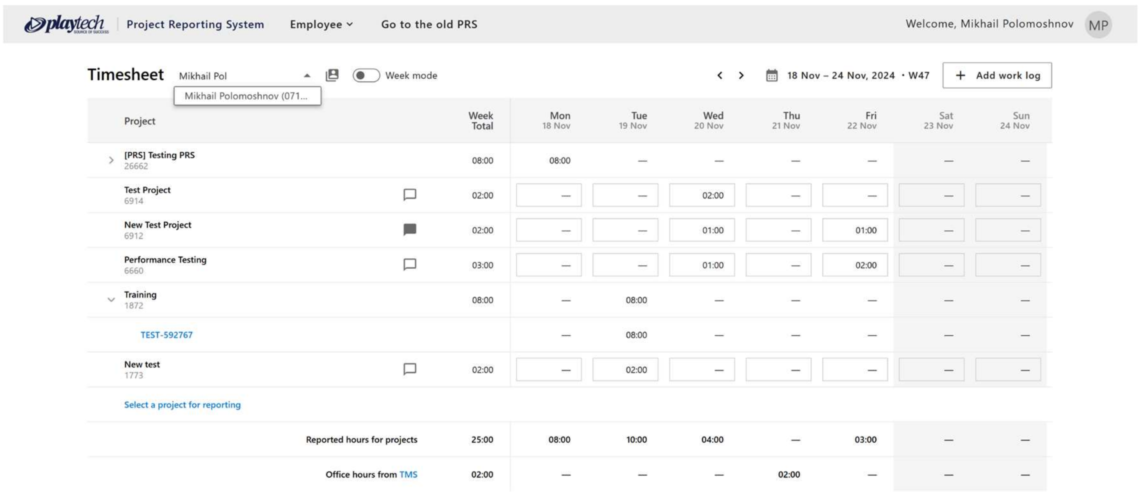
Jul, 2024

Mon	Tue	Wed	Thu	Fri	Sat	Sun
1 08:00/08:00	2 08:00/08:00	3 08:00/08:00	4 08:00/08:00	5 00:00/08:00	6 00:00/08:00	7 00:00/08:00
8 08:00/08:00	9 08:00/08:00	10 08:00/08:00	11 08:00/08:00	12 08:00/08:00	13 00:00/08:00	14 00:00/08:00
15 08:00/08:00	16 08:00/08:00	17 08:00/08:00	18 08:00/08:00	19 08:00/08:00	20 00:00/08:00	21 00:00/08:00
22 08:00/08:00	23 08:00/08:00	24 08:30/08:00	25 08:00/08:00	26 08:00/08:00	27 00:00/08:00	28 00:00/08:00
29 08:00/08:00	30 08:00/08:00	31 08:30/08:00				

24 July	08:30 / 08:00
[Test] New Test 2024 25669	04:00
[Test] Test Project 2024 25706	04:00
TEST-7456	04:00
Test Project 27384	00:30

Figure 9. Timesheet view, month mode.

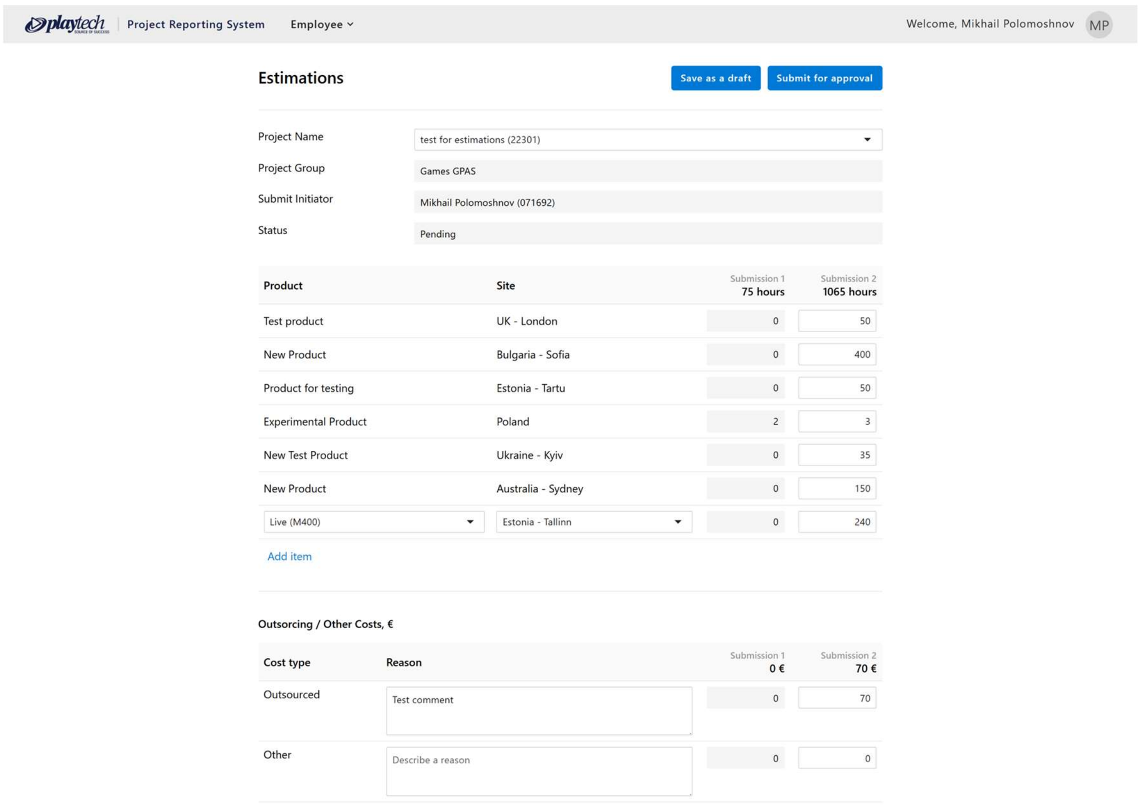
Appendix 8 – Timesheet view, week mode



Project	Week Total	Mon 18 Nov	Tue 19 Nov	Wed 20 Nov	Thu 21 Nov	Fri 22 Nov	Sat 23 Nov	Sun 24 Nov
[PRS] Testing PRS 26662	08:00	08:00	—	—	—	—	—	—
Test Project 6914	02:00	—	—	02:00	—	—	—	—
New Test Project 6912	02:00	—	—	01:00	—	01:00	—	—
Performance Testing 6660	03:00	—	—	01:00	—	02:00	—	—
Training 1872	08:00	—	08:00	—	—	—	—	—
TEST-592767		—	08:00	—	—	—	—	—
New test 1773	02:00	—	02:00	—	—	—	—	—
Select a project for reporting								
Reported hours for projects	25:00	08:00	10:00	04:00	—	03:00	—	—
Office hours from TMS	02:00	—	—	—	02:00	—	—	—

Figure 10. Timesheet view, week mode.

Appendix 9 – Estimations view



Estimations

Project Name	test for estimations (22301)
Project Group	Games GPAS
Submit Initiator	Mikhail Polomoshnov (071692)
Status	Pending

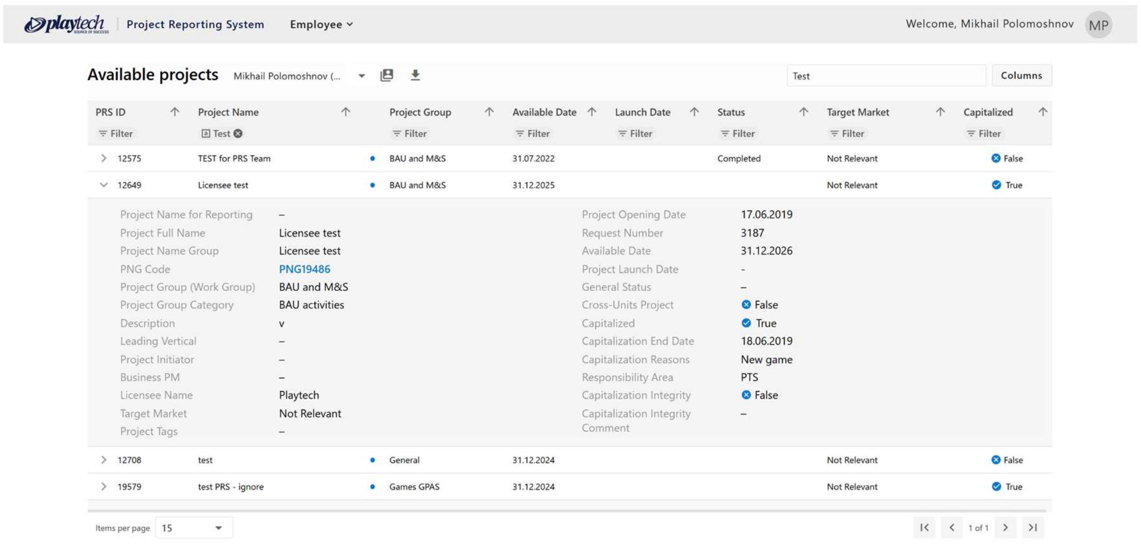
Product	Site	Submission 1 75 hours	Submission 2 1065 hours
Test product	UK - London	0	50
New Product	Bulgaria - Sofia	0	400
Product for testing	Estonia - Tartu	0	50
Experimental Product	Poland	2	3
New Test Product	Ukraine - Kyiv	0	35
New Product	Australia - Sydney	0	150
Live (M400)	Estonia - Tallinn	0	240

Outsorcing / Other Costs, €

Cost type	Reason	Submission 1 0 €	Submission 2 70 €
Outsourced	Test comment	0	70
Other	Describe a reason	0	0

Figure 11. Estimations view.

Appendix 10 – Available projects view



Available projects — Mikhail Polomoshnov (...

Test | Columns

PRS ID	Project Name	Project Group	Available Date	Launch Date	Status	Target Market	Capitalized
Filter	Test	Filter	Filter	Filter	Filter	Filter	Filter
12575	TEST for PRS Team	BAU and M&S	31.07.2022		Completed	Not Relevant	False
12649	Licensee test	BAU and M&S	31.12.2025			Not Relevant	True
12708	test	General	31.12.2024			Not Relevant	False
19579	test PRS - ignore	Games GPAS	31.12.2024			Not Relevant	True

Details of 12649:

Project Name for Reporting	–	Project Opening Date	17.06.2019
Project Full Name	Licensee test	Request Number	3187
Project Name Group	Licensee test	Available Date	31.12.2026
PNG Code	PNG19486	Project Launch Date	-
Project Group (Work Group)	BAU and M&S	General Status	–
Project Group Category	BAU activities	Cross-Units Project	False
Description	v	Capitalized	True
Leading Vertical	–	Capitalization End Date	18.06.2019
Project Initiator	–	Capitalization Reasons	New game
Business PM	–	Responsibility Area	PTS
Licensee Name	Playtech	Capitalization Integrity	False
Target Market	Not Relevant	Capitalization Integrity Comment	–
Project Tags	–		

Items per page 15 — 1 of 1

Figure 12. Available projects view.

Appendix 11 – The old application performance testing results

Table 3. The old application performance testing average results.

View	**First contentful paint, (s)**	**JavaScript execution time, (s)**	**Speed index, (s)**	**Total blocking time, (s)**
New project request	4.7	1.1	6.1	0.52
Timesheet	4.6	1.1	7	0.61
Estimations	4.7	1	6.7	0.76
Available projects	4.8	0.8	6.2	0.5

Appendix 12 – The new application performance testing results

Table 4. The new application performance testing average results.

View	**First contentful paint, (s)**	**JavaScript execution time, (s)**	**Speed index, (s)**	**Total blocking time, (s)**
New project request	0.9	0.3	1.9	0.03
Timesheet	1.1	0.3	1.8	0.04
Estimations	0.9	0.1	1.6	0.01
Available projects	1.2	0.4	2.8	0.09