Requirements and Analysis Document for Hyro

TDA367, Chalmers tekniska högskola

Eimer Ahlstedt, Emil Lindblad, Sebastian Kvaldén, Timothy Nilsson, Erik Larsson

24 oktober 2021

Table of contents

1 Introduction

The purpose of this application is to enable people to participate in sharing economy, by allowing users to put their own items up for rent as well as rent other users' items. This is achieved by creating a virtual marketplace, where users can create an account, browse listings and also add their own items for rent.

People who can benefit from this application can be divided into two categories: the ones who rent items and the ones who supply items for rent.

Imagine that you have just bought some new shelves and realizing that you do not have a hammer drill for mounting them to your concrete wall. Instead of spending a lot of on a new one or leasing one with improper and complicated agreements, you can just check if anyone nearby has a hammer drill for rent on Hyro.

In the other case, you might have a tool or item that you only use a few times a year, like a pressure washer or a trailer. Instead of letting these items collect dust in your garage, you can rent them out on Hyro. Making them available for more people, decreasing their environmental footprint.

1.1 Definitions, acronyms and abbreviations

- **Jira** - A tool used for Agile software management.
- **JSON** - JavaScript Object Notation. A human readable data format usually used for data interchange with webservers. We use it in our application for storing and persisting data between sessions.
- **JavaFX** - A Java framework used for creating graphical users interfaces.
- **GitHub** - A tool used for storing code repositories.
- **FXML file** - A type of file used by JavaFX. Contains information on visual elements and what code is connected to each element.
- **MVC design pattern** - A design pattern of object oriented programming. The goal of the pattern is to divide a code base into three parts, one Model, one View and one Controller, each one with different responsibilities.

2 Requirements

2.1 User Stories

The following user stories is implemented or should be implemented in the future.

2.1.1 Runnable Program

Status: DONE

Description
As a user I want to be able to run the program, so that I can start using it.

Confirmation

Functional:
Program compiles and basic window opens.

Non-Functional:
Project created and accessible through GitHub.
Configure maven and JavaFX.

2.1.2 Data Storage

Status: DONE

Description
As a User I want to be able to see the same information when I restart the application so that I can continue renting later.

Confirmation
Functional:
A User should be able to see the same listings after restarting the application.

Non-Functional:
JSON-files created and tested for User and Listing-classes.
Fully functional read/write-methods to access the JSON-files

2.1.3 Log-In Screen and Functionality

Status: DONE

Description
As a User, I want to be able to enter my login credentials, so that I can be logged in.

Confirmation
Functional:
Accessible Log-In page for users to enter their credentials or create account.

Non-Functional:

2.1.4 Views and Navigation Buttons

Status: DONE

Description
As a User I want to be able to navigate through the different pages, so that I can explore the application.

Confirmation
Functional:
Access different views as a user. Such as Log-In and Settings screen.

Non-Functional:

2.1.5 User Settings

Status: DONE

Description
As a User I want to be able to change my personal information (address, password etc).

Confirmation
Functional:
View that shows and enables changes to the logged in user's information.

Non-Functional:
Check for correct syntax in user input and logged in user.

2.1.6 Grid of Listings and Detail View

Status: DONE

Description
As a User I want to be able to view a product, so that I can determine if I want to rent it.

Confirmation
Functional:
A user should be able to view a pre-defined product in the application.

2.1.7 Add listings page and functionality

Status: DONE

Description
As a User I want to be able to add a product for rent. So that I can show other users that I have products for rent.

Confirmation
Functional:
Logged-in user can create a product and let other users rent it.

2.1.8 Create Account Page

Status: DONE

Description
As a User, I want to be able to create an account so that I can keep track of my data and log in.

Confirmation
Functional:
Create Account-button on log-in page for new users to create account.

2.1.9 Rent-Button

Status: DONE

Description
As a User I want to be able to request to rent a product, so that can I show the owner that I am interested.

Confirmation
Functional:
A logged-in user can request to rent a product that another user has advertised as available.

2.1.10 Browse View for Listings

Status: DONE

Description
As a User, I want to be able to see a list of all products that are available for renting, so that I can browse and decide what I want to rent.

Confirmation
Functional:
A main page where a user can browse products.

2.1.11 Sort and search

Status: DONE

Description
As a User I want to be able to sort and search for products that are available.

Confirmation
Functional:
Search-function is usable and shows relevant products. Category buttons changes shown products to match the specific category.

2.1.12 Log-Out Button

Status: DONE

Description
As a User I want to be able to log-out of the application when i am done using it.

Confirmation
Functional:
Functioning log-out button that takes a user back to log-in screen.

2.1.13 Edit Listings

Status: DONE

Description
As a User I want to be able to edit my already existing listings so that i can supply the correct and latest information.

Confirmation
Functional:
Functional page to see the users current listings and be able to change their information.

2.1.14 Images

Status: DONE

Description
As a User I want to be able to upload a picture of my listing, so other users can get a better understanding of my offer.

Confirmation
Functional:
Pictures are shown with listing objects.

2.1.15 Remove Listings

Status: DONE

Description
As a User I want to be able to remove my listings so they cant be rented anymore.

Confirmation
Functional:
Remove Objectbutton at detail view for my listing. Listings that are already out for rent can´t be removed, however available objects are remove able.

2.1.16 Remove Completed Bookings

Status: DONE

Description
As a User I want to be able to remove my completed bookings, so that they don't show anymore.

Confirmation
Functional:
Remove-button at bookings i have previously made to remove them from my list.

2.1.17 Book in specific time frames

Status: NOT IMPLEMENTED

Description
As a User I want to be able to rent a product during a specific date range, so that i can specify when i want to rent an item.

Confirmation
Functional:
User has the ability to book a product for a future date.

2.1.18 Decline bookings

Status: NOT IMPLEMENTED

Description
As a User I want to be able to decline a booking request, so that I don´t have to rent my product.

Confirmation
Functional:
Decline button for the owner of the product to use if they don´t want to rent it out.

2.1.19 Cancel booking requests

Status: NOT IMPLEMENTED

Description
As a User I want to be able to cancel my booking request, so that i don´t have to follow through with a booking i don´t want anymore.

Confirmation
Functional:
The user has the ability to cancel a booking request that they made before it is accepted by

the product owner.

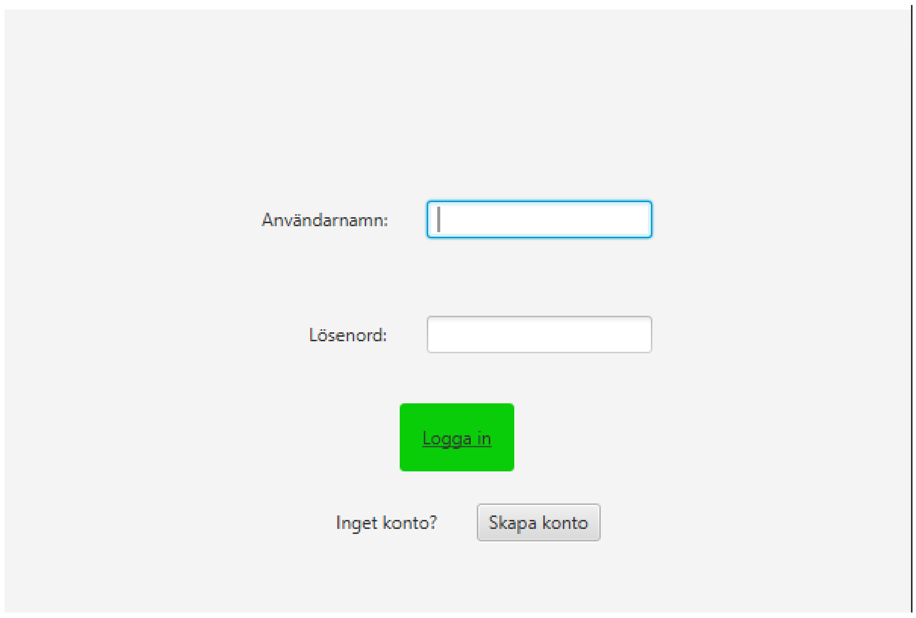
2.2 Definition of Done

New features are worked on in a new branch

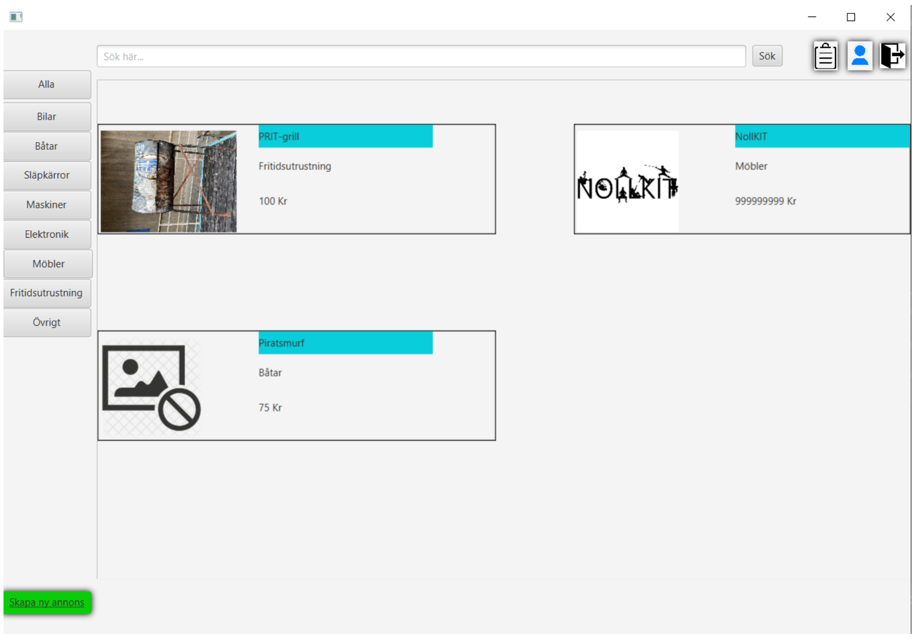
Public parts of code are documented.
90% Test Coverage of relevant code in Model classes. Simple getters and setters and states are considered irrelevant for tests. All checks pass and merge to master is successful

2.3 User interface

The User Interface is created using JavaFX and consists of multiple FXML files. Each view that the user can navigate to is a Scene which is where all the data gets passed through, both input and the output. Each Scene has its own controller that is in charge of changing the scene that user wants to see. So if for example a users presses Log in button and all log in checks passes, the controller will request the SceneHandler to then switch the scene the browse view. The Scene Handler then loads the respective FXML file and then displays it in the application.



Figur 1: Login in view where users can log in to an existing account or create a new one

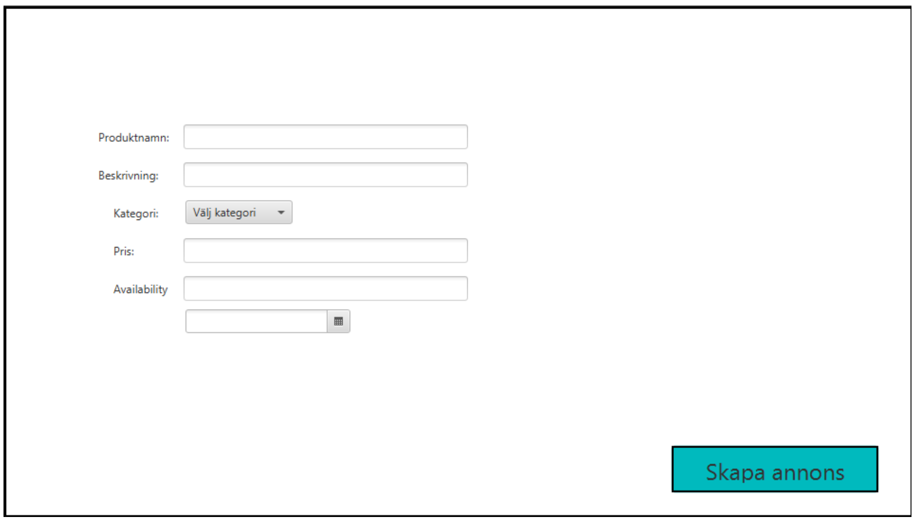


PRIT-grill
Fritidsutrustning
100 Kr

NollKIT
Möbler
999999999 Kr

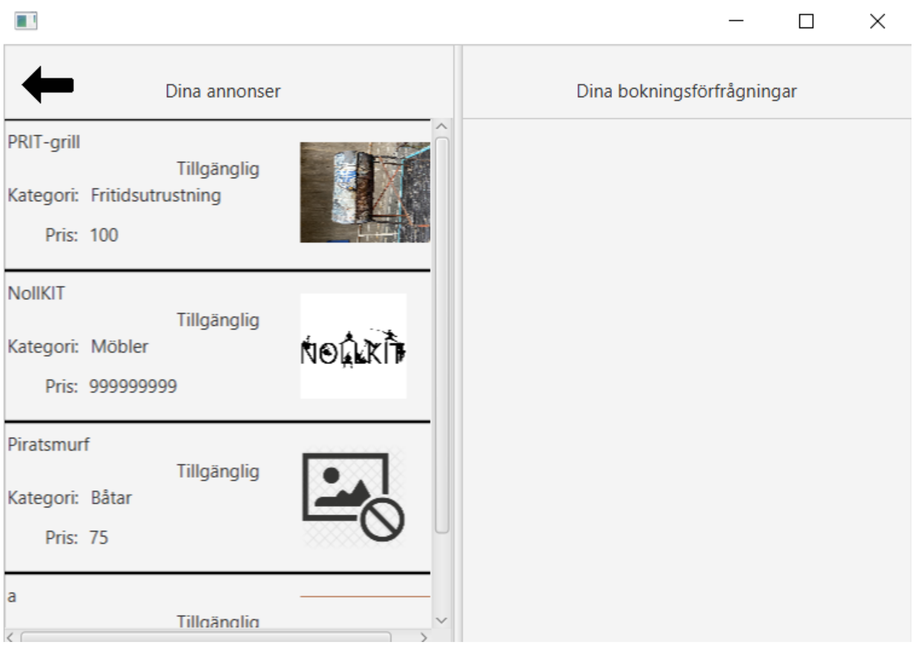
Piratsmurf
Båtar
75 Kr

Figur 2: Browse view where logged in users can see available listings for rent and also create a new listing



Produktnamn:
Beskrivning:
Kategori: Välj kategori
Pris:
Availability

Figur 3: The standardized view for listing settings, used both to create and edit listings.


Dina annonser

Dina bokningsförfrågningar

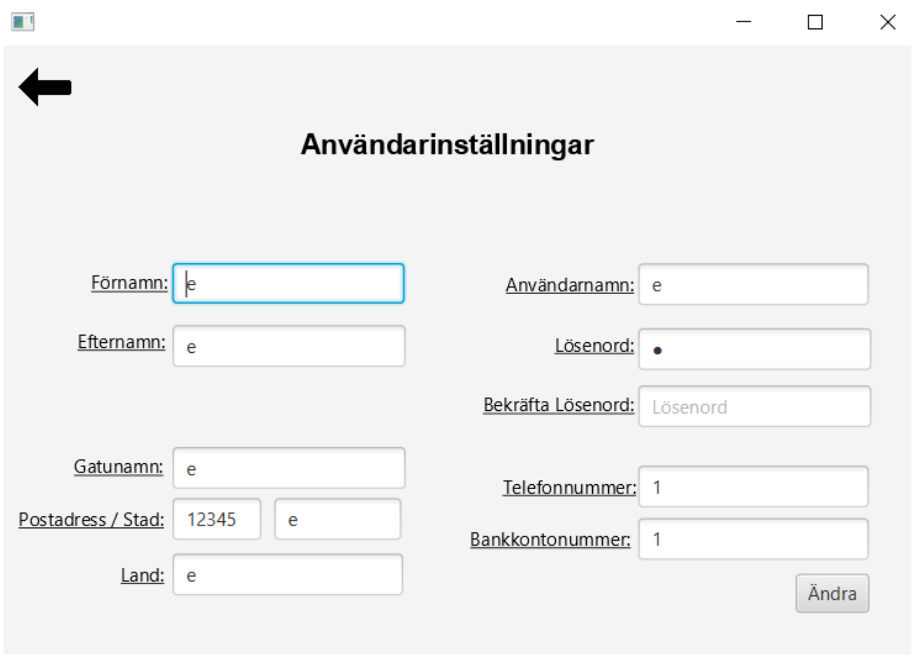
PRIT-grill
Tillgänglig
Kategori: Fritidsutrustning
Pris: 100

NollKIT
Tillgänglig
Kategori: Möbler
Pris: 999999999

Piratsmurf
Tillgänglig
Kategori: Båtar
Pris: 75

a
Tillgänglig

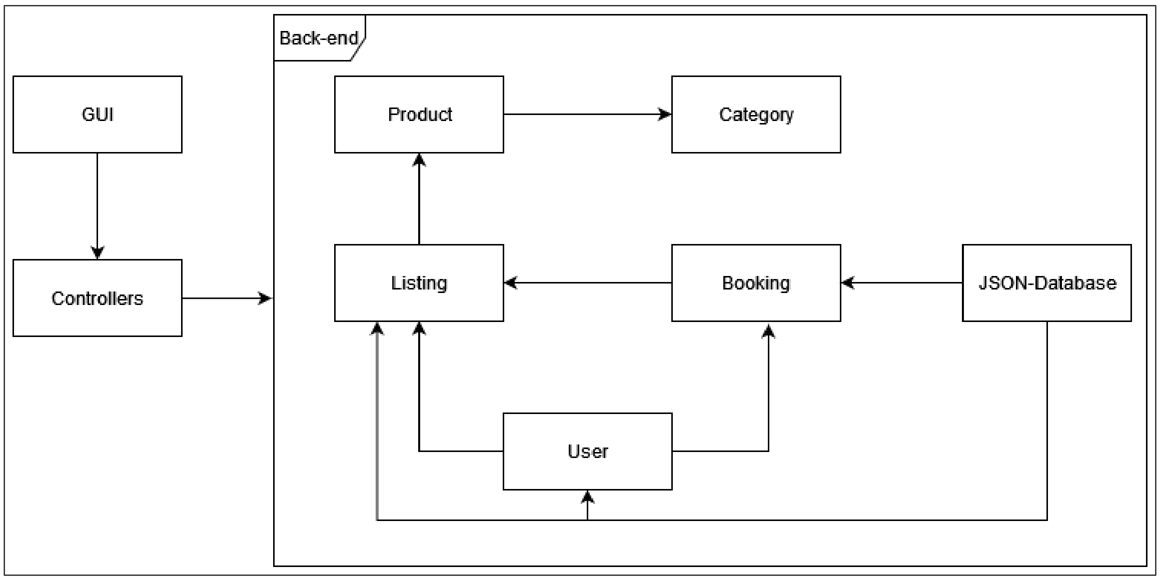
Figur 4: View for a users own listings and active booking requests. Also used for accepting booking requests and making payments.


Användarinställningar

Förnamn:	e	Användarnamn:	e
Efternamn:	e	Lösenord:	•
		Bekräfta Lösenord:	Lösenord
Gatunamn:	e	Telefonnummer:	1
Postadress / Stad:	12345 / e	Bankkontonummer:	1
Land:	e		

Figur 5: The standardized view for account settings, used both to create and edit accounts.

3 Domain Model



Figur 6: The current Domain Model of Hyro

3.1 Class responsibilities

Above is the current state of the Hyro Domain Model.

The program is built around the MVC design pattern. The class ”GUI” contains all FXML files and their controller classes, making up the View part. FXML controller class is not to be confused with MVC Controller; FXML controller classes are simply the Java classes connected to FXML files.

”Controllers” represents the MVC Controller classes. View classes chain calls to Controllers, who in turn update the View with information from the Back-end, or Model.

The Back-end, or Model, contain the logic and data needed in the program.

The User class is for keeping track of an individual users contact information, currently listed products and active booking requests.

Listing exists to contain information on what product has been listed, and by which user. It also keeps track of the current state of the listing such as if the listing is available or has a booking request etc etc.

The purpose of Booking is to be the middle ground between a customer wanting to rent a product and the owner of the product. It keeps track of the state of the booking and controls the process of booking, paying and returning.

Product and Category are simple. A product is the item available for rent, and the Product class contains information on the item, such as its category. The Category is used to filter searches, for example if a user wants to rent any grill, it could browse a hypothetical ”grill”-category.

References

NA