November 28, 2015 v1.04

Team C

PostCardBuddy

System Requirements

Authors of this document:
Emma Albertz
Caroline Brandberg
Linnéa Claesson
Billy Johansson
Johan Ju
Jacob Mejvik
Carl Rynegardh

Contents

1 Introduction

This document is written within the context of the course Requirements Engineering at Lund Institute of Technology, which the authors are currently enrolled in. They have been provided with a project mission from another group, specifying a product they want to see developed. The intention of this document is to specify the requirements of this product, namely PostCardBuddy.

2 Background

Why does nobody send postcards? Because the process is tedious and takes effort - but not any more! With PostCardBuddy it will be easy to send postcards to everyone you know. PostCardBuddy is a mobile application that will simplify the whole process, whether you want to be creative and design your own postcards or make it easy for yourself and use a template postcard based on your location and send it to everyone in your contact list.

The application is perfect for every occasion you want to send a postcard. Grandma's birthday is coming up? Send a postcard of you and your cousins! Christmas is around the corner? Send everyone in your contact list a postcard of your cats! Away on vacation? Why not send a ready-made postcard that shows off the amazing beach to everyone in the office? Nobody needs to know it rained all week.

PostCardBuddy is the perfect tool when you want to let someone know you are thinking of them.

3 System Requirements

3.1 Goal

The product aims to increase the amount of postcards sent and shall achieve this through the following goals:

- Simplify the process of sending postcards
- Enable user to send personalized postcards
- It shall be possible to generate revenue through the system

3.2 Domain

3.2.1 Tasks

3.2.2 Supported systems

Req 1.2.1.1 The system shall support iOS.

Req 1.2.1.2 The system shall support Android.

Req 1.2.1.3 The system shall support a mobile payment solution.

3.2.3 Interfaces

Req 1.2.2.1 The interface connecting PostCardBuddy and the printed postcards is an off-the-shelf printer.

Req 1.2.2.2 The system sends image files to a printer.

3.3 Product

Req 1.3.1.1 Success notification The user shall be notified when an order is sent from a device.

Req 1.3.1.2 Fail notification The user shall be notified when an order fails to be sent from a device.

Req 1.3.1.3 No internet If the user places an order on a device that is not connected to the internet, the order shall be stored and sent the next time the device receives internet connection.

3.4 Design

Req 1.4.1.1 Front page The front of the postcard shall be a field containing an image.

Req 1.4.1.2 Text field The back of the postcard shall contain a text field.

Req 1.4.1.3 Address field The back of the postcard shall contain an address field.

Req 1.4.1.4 Postage field The back of the postcard shall contain a postage field.

Req 1.4.1.5 Postage print The postage shall be printed in the top right corner on the back of the postcard.

Req 2.4.1.5 Start Screen The application shall start with a screen where its possible to choose front and back image/text figure 1 upper left.

Req 2.4.1.6 Get image The application shall let the user choose the image source from a menu 1 upper right.

Req 2.4.1.7 Edit image The application shall give the user a basic image editor to customize the image 1 lower left.

Req 2.4.1.8 Recipiant address The application shall have an address input screen with an address-book/contacts (not in image) 1 lower right.

3.5 Data Requirements

Req 1.5.1.1 Data model The system shall handle the data presented in the data model in figure 2.

3.5.1 Data dictionary

Class: Name
Description what it is exactly, what it does and what it interacts with

Examples:

1. What is this typically?
2. Special cases.

Attributes:

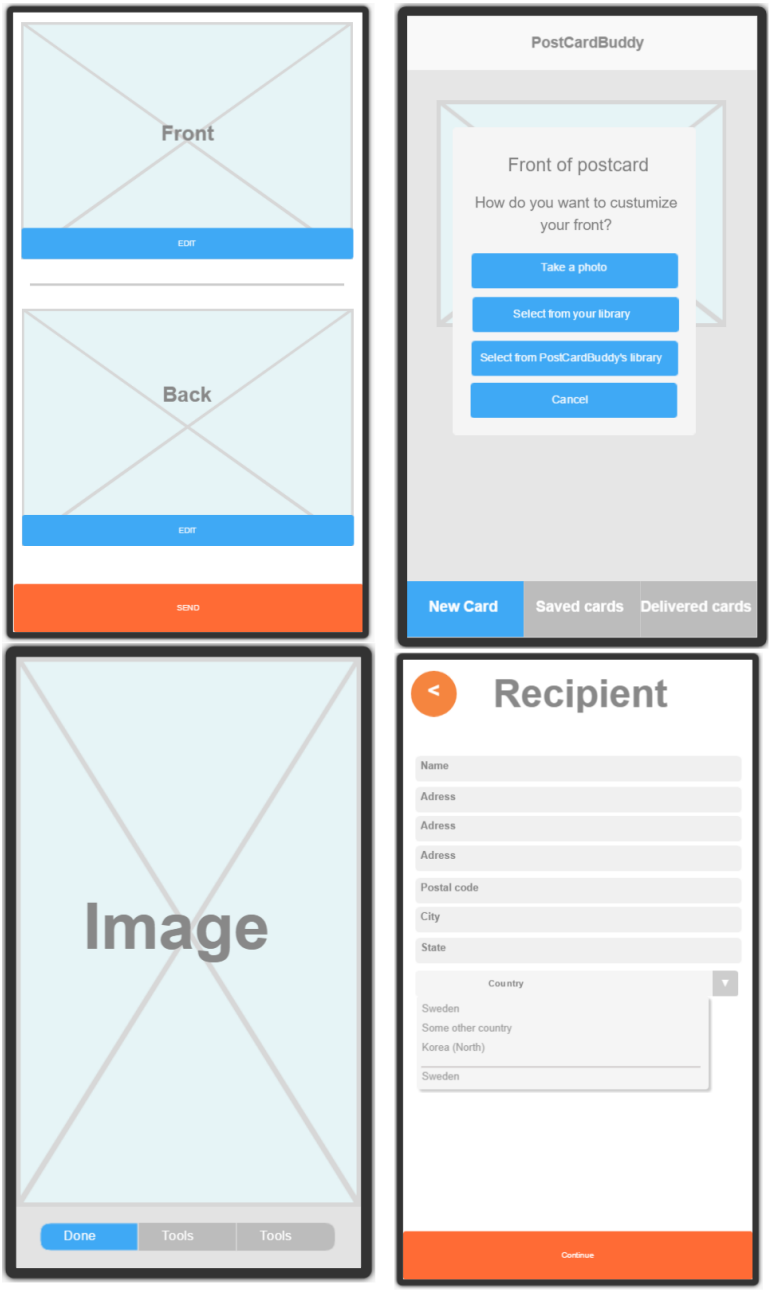

Figure 1: The prototype

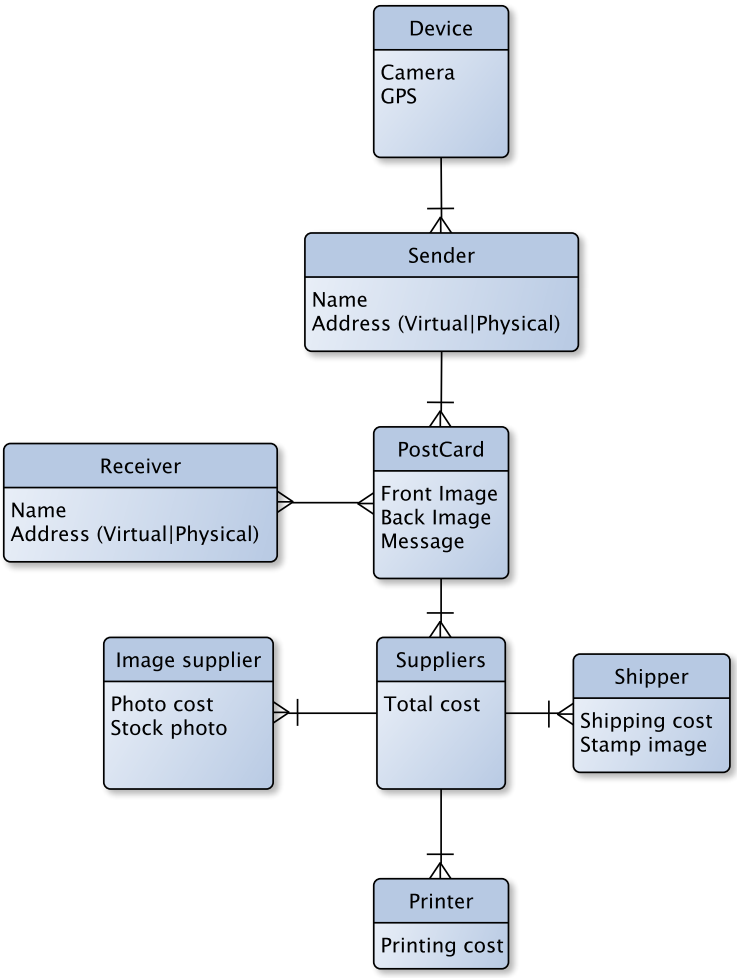

Figure 2: Data model

1. List attributes and
2. What type of data it is

Class: Device

The device is the actual physical mobile device on which the application is running.

Examples:

1. An android device running the application.
2. An iOS device running the application

Attributes:

1. **Camera:** Image
   A compressed image fetched from the devices physical camera.
2. **GPS:** String[Latitude,longitude]
   The information about current coordinates from the GPS in the device. The string is given on the format shown and the latitude and longitudes are signed floats with seven decimal places.

Class: Sender

This class represents the person sending the post card. It can be the same person as the one using the device but it doesn't have to.

Examples:

1. The device owner.
2. A person using the application to send a post card.

Attributes:

1. **Name:** String
   The name of the sender.
2. **address** (Virtual—Physical): String
   This attribute is always a string. If it's a virtual address it's a user name, otherwise it's a physical street address.

Class: Receiver

This class represents the person receiving the post card. This class is identical to *Sender* in terms of attribute structure. The sender and receiver could be the same person.

Examples:

1. The person receiving the post card.
2. The same person as the one sending a post card.

Class: PostCard

This class represents the the post card sent from the *Sender* to *Receiver*. It encapsulates all the information necessary to send a post card in either virtual or physical form. An instance of this object, owning a *Sender* and a *Receiver* needs to exist to be able to send a Post Card.

Examples:

1. A post card with two images, a message and a stamp.
2. A post card with no images, no message and a stamp.
3. A virtual post card with images, a message and no stamp.

Attributes:

1. **Front image:** Image [optional]
   A compressed image that will be used as the front image of the *PostCard*.
2. **Back image:** Image [optional]
   A compressed image that will be used as the back image of the *PostCard*.
3. **Message:** String [optional]
   The message on the *PostCard*.
4. **Stamp image:** Image
   The image supplied by *Shipper* to properly send the post card.

Class: Suppliers

This class collects the data from an *Image supplier*, a *Shipper* and a *Printer*.

Examples:

1. A collection of suppliers relevant to printing and sending a specific *PostCard*.
2. Only a printing and shipping cost.

Attributes:

1. **Total cost:** Float
   The combined cost of *Image supplier/Photo cost*, a *Shipper/Shipping cost* and a *Printer/Printing cost*. The value is rounded up to two decimal places.

Class: Image supplier:

This class represents a supplier of images. If a user chooses a stock photo as (for example) a *PostCard/Front image* there is a cost with using the photo that needs to be added to the total cost.

Examples:

1. An image supplier with a photo and a cost.
2. An image supplier with a free photo.

Attributes:

1. **Photo cost:** Float
   The cost of a stock photo. The value is rounded up to two decimal places.
2. **Stock photo:** Image
   The actual image that will be bought.

Class: Shipper:

This class represents a shipper. The shipper is the company that will transport the post card.

Examples:

1. A representation of what is required to send a post card with Posten.
2. A representation of what is required to send a post card with DHL.

Attributes:

1. **Shipping cost:** Float
   The cost of shipping. The value is rounded up to two decimal places.
2. **Stock photo:** Image
   This is the image used on the post card to indicate that shipping was payed for.

Class: Printer:
This class represents a printer. The printer is repsponsible for printing the physical post card.

Examples:

1. A company contracted to print a post card.
2. The company supplying the application.

Attributes:

1. **Printing cost:** Float
   The cost of printing. The value is rounded up to two decimal places.

3.5.2 Virtual windows

Req 1.5.1.2 PostCard The input data to the *PostCard* class described in the Data dictionary shall include the items specified in the virtual window in figure 3.

Req 1.5.1.3 Sender The input data to the *Sender* class described in the Data dictionary shall include the items specified in the virtual window in figure 4.

Req 1.5.1.3 Sender The input data to the *Receiver* class described in the Data dictionary shall include the items specified in the virtual window in figure 5.

3.6 Functional Requirements

3.6.1 Images

Req 3.6.1.1 Image from gallery It shall be possible to choose pictures from the phone gallery for the front of the postcard.

Req 3.6.1.2 Picture from camera It shall be possible to take a picture through the camera and use as image for the front of the postcard.

Req 3.6.1.3 Image and GPS position It shall be possible to choose images depending on the user's GPS position.

Req 3.6.1.4 Image editing The system shall have a function for editing of images.

Req 3.6.1.5 Image saving It shall be possible to save images.

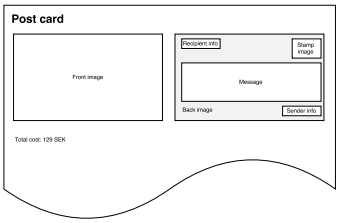

Figure 3: Virtual window PostCard

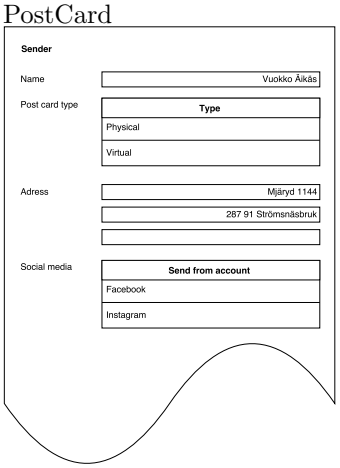

Figure 4: Virtual window Sender

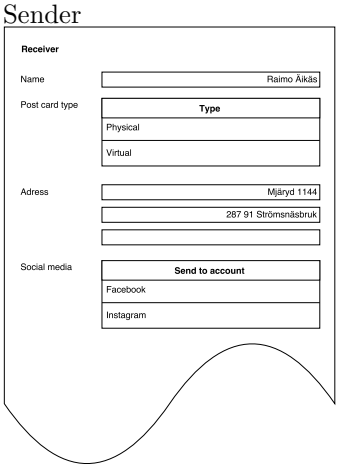

Figure 5: Virtual window Receiver

3.6.2 Greetings

Req 3.6.2.1 Greetings It shall be possible to write greetings in the app.

Req 3.6.2.2 Pictures of handwritten greetings It shall be possible to choose a picture of

a hand-written greeting.

Req 3.6.2.3 Auto-generated greetings It shall be possible to choose a template greeting.

Req 3.6.2.4 GPS based greetings The system shall be able to generate greetings based on GPS position.

Req 3.6.2.5 Handwritten greetings on screen It shall be possible to write a handwritten greetings directly on the screen.

Req 3.6.2.6 Saving greetings It shall be possible to save a greeting.

3.6.3 Recipients

Req 3.6.3.1 Enter recipients It shall be possible to enter recipients manually.

Req 3.6.3.2 Phone book recipients It shall be possible to choose recipients through the phone book.

Req 3.6.3.3 Multiple recipients The system shall be able to handle multiple recipients for one postcard.

Req 3.6.3.4 Favourite recipients It shall be possible to save recipients as favourites.

Req 3.6.3.5 Frequent recipients The system shall show frequently used recipients as favourite recipients.

3.6.4 Postcard

Req 3.6.4.1 Saving postcards It shall be possible to save postcards.

Req 3.6.4.2 Reuse postcards It shall be possible to reuse saved postcards.

Req 3.6.4.3 Digital postcard It shall be possible to send digital postcards.

Req 3.6.4.4 Physical postcards It shall be possible to send physical postcards.

Req 3.6.4.5 Postcard size It shall be possible to choose the size of the physical postcard.

Req 3.6.4.6 Quality of physical postcard It shall be possible to choose the print quality of physical postcards.

Req 3.6.4.7 History It shall be possible to display the history of sent postcards.

Req 1.6.3.8 Social media Feature for sharing postcards on social media.

3.7 Quality Requirements

3.7.1 Performance

Req 3.7.1.1 Memory usage The application shall adjust its memory usage depending on the device.

Req 3.7.1.2 Speed The user interface shall be considered smooth on devices faster than Nexus 5 / iPhone 5 for 8 out of 10 users.

Req 3.7.1.3 Picture quality The camera shall be able to take a picture in the highest hardware supported resolution.

Req 3.7.1.4 Autofocus The camera shall have a autofocus that is comparable to the Android / iOS stock camera.

3.7.2 Availability

3.7.3 Security

Req 3.7.3.1 Store cards The photos shall be stored encrypted

Req 3.7.3.2 Sending card The photos shall be sent encrypted to back-end.

3.7.4 Maintainability/Portability

Req 3.7.4.1 Language The application shall be developed in non native language e.g. Java for Android.

Req 3.7.4.2 Device support The application shall work on devices with newer operating systems than Android 4.1 / iOS 7.0.1

3.7.5 Usability

Req 3.7.5.1 User friendly 9 out of 10 users shall be able to use the system after a five minute instruction.

4 Specification Techniques

4.1 Context Diagrams

Figure 6 shows the context diagram. This diagram shows the interface that the application PostCardBuddy will interact with and the stakeholders who will interact with the application.

4.2 Features

4.3 Virtual Windows

4.4 Task Descriptions

5 Release Plan

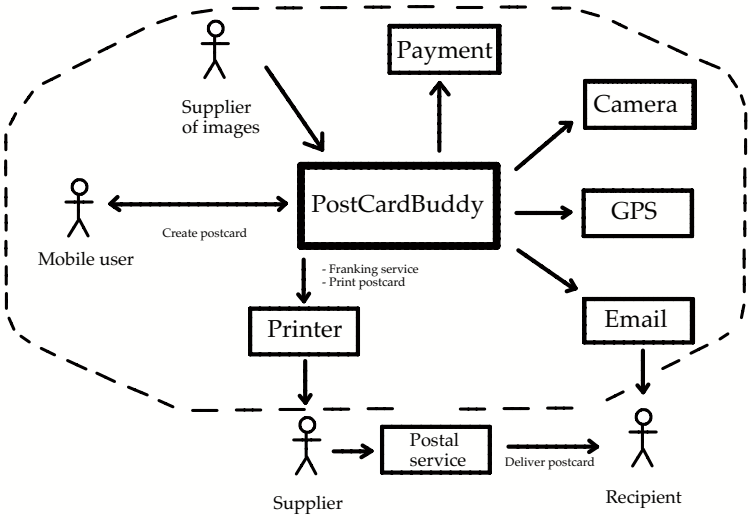

Figure 6: Context diagram of product.