Account This!

Group 9

Johannes Edelstam
Joakim Ekberg
Kristoffer Renholm
Jesper Skoglund

Preface

This requirements document intends to describe the raison d'être as well as the underlying logic of Account This!, a novel bookkeeping solution with several advantages to the many static alternatives often used in today's world of business. The system aims to treat a point of pain seen among many small enterprises and can, due to it's ease of use, be expected to delight a broad variety of customers.

The document is intended for he or she who is interested to learn about the rationale behind Account This!. The text depicts what the system does and by whom it is intended to be used. A thorough list of requirements, staking out the limits and the possibilities of the system, is shown and explained. All this are supported by motives and models to further enhance readers' likelihood of fully grasping the system. We, the team behind the Account This!, are most grateful for the many advices we've received from our supervisor, Professor Rand Waltzman of The Royal Institute of Technology, whose help and support has been invaluable during each stage of the project.

Stockholm, December 20, 2007

Table of Contents

Preface 2

Introduction 5

Who are the users and what problem does the system solve for them? 5

The main uses of the system 5

Use case 1 5

Use case 2 5

The context/environment in which the system is to be used 5

The scope of the system 6

The main factors that need to be taken in to account when designing and building the system 6

Technologies and Risks 7

Glossary 8

User requirements 9

Non-functional 9

Security 9

Usability 9

Reliability 9

Standards 10

Availability 10

Performance 10

Scalability/extensibility 10

Quality 10

Compability 10

Functional 11

Users 11

Companies 11

Fiscal years 11

Vouchers 11

Support 11

Accounting plans 11

Use cases 12

Users 12

Fiscal year 17

Vouchers 18

Voucher rows ... 19
Edit a saved voucher row ... 20
Accounting plan ... 21
Support ... 25
Product standards ... 25
System requirements ... 26
Functional ... 26
Users ... 26
Companies ... 26
Fiscal years ... 26
Vouchers ... 27
Accounting plans ... 27
Support ... 27
System architecture ... 27
Overview ... 27
Model-View-Controller architecture ... 27
System evolution ... 28
Appendices ... 28
Hardware ... 28
Client ... 28
Server ... 28
References ... 28

Introduction

Who are the users and what problem does the system solve for them?

Account This! aims to deliver a bookkeeping system suitable for small companies. Such companies could for example be hairdressers (almost every hairdresser in Sweden has a one-man-company which rents a hairdressers chair in a barbershop), consultants or similar small companies. Small companies often have small budgets where every cent matter. Furthermore, small companies do not have a lot of knowledge in bookkeeping; therefore, they need a bookkeeping system which is easy to use (a system that comes with a high degree of usability and includes all essential bookkeeping functions). We have discovered that very few products are able to satisfy this need. Small companies need to commit their resources to the creation of value (by focusing on the core activities of the business), not on bookkeeping. This pose a potential market for a cheap and effective bookkeeping system such as the one proposed; Account this!.

The main uses of the system

Account This! will be a system for double-entry booking; the standard used in most businesses and organizations. Unlike other, often large and complex, bookkeeping systems, ours will be web-based, easy to access and easy to use. Account This! should thus be able to help its intended users (mainly small companies) do, view and edit their bookkeeping. To reflect the need of these, the system must be quick to learn and easy to use. This implies that Account This! will only cover the most essential functions of bookkeeping, and that the system may not be fully suitable for larger companies.

Use case 1

Peter runs a small painting firm. He has just received payment for a job. He logs into Account This! by stating a username, a password and pressing the Login button. Peter clicks “New voucher” and a table appears. He decides to add a row to the table and does so by entering information into a set of neatly places cells and pressing the “Add row” button. This debts one account of his choice and automatically generates another row which credits another account. The task is then completed by a click on the “Create” button. Peter feels content and is once again surprised by how little time it took to complete the assignment with Analyze This!.

Use case 2

Anna, the CFO of a small commodity sales firm, has been entering information into her firm’s Analyze This! bookkeeping table when she suddenly realizes that she’s made a fatal mistake; the sum of an invoice she added a few minutes ago was incorrect and must be corrected. Anna navigates the mouse icon to the frame showing the functionality of the system. Instead of using the search opportunity, she clicks the “Voucher History” button where she finds the sought voucher by looking at the IDs and timestamps seen in the vouchers list. Due to Swedish law Anna would be unable to edit the voucher, so instead of changing already entered information, the system corresponds to Anna’s will by creating a new voucher which overrides the old one. Anna will thus, due to the functionality of the system, feel as if she actually did edit the voucher.

The context/environment in which the system is to be used

From the users’ point of view, the system will be accessed directly through his/her web-browser on an internet connected computer (please refer to chapter six for more information concerning system technology). In this context the only requirements of the environment should be that the user uses a modern web-browser. In our case, a modern web browser is defined as either one of the following:

- Firefox 2.0
- Internet Explorer 7
- Safari 3

The limitation to three browsers is a result of the very complicated testing procedures that otherwise could occur. If a user however decides to use an alternative web-browser (such as Opera) nothing will hinder him/her to try. The only implication of such a move is that we, as developers and designers, won't be able to guarantee that the system functions as is intended. Analyze This! can hence be used from a huge variety of environmental locations (from the workplace, at the library, at home etc.). Consequently, it's very hard to specify any general criteria stemming from the ethnography of the users and from other macro contextual aspects.

The scope of the system

To make the Account This! project successful we need to meet more goals than what's possible to do in this course. This is due to time restrictions and manpower. We would need much more time of development. Because of this we just select the most important subset of features to get a great prototype.

We want do develop the most basic accounting application there is and put it online. The prototype will support the most basic features of an accounting application but will be built in a manner that makes it simple to extend. These features are presented below in an in-out chart.

Topic	In	Out
Add voucher	X	
Handle the way a voucher is changed according to all laws that apply	X	
Manage different accounts	X	
Able to import the 'bas' - chars of accounts.	X	
Import data from other systems according to the standard SIE-format. This format could easy be exported from, for example, the well known SPCS Accounting Program.	X	
Generate different types of reports for example Balance-Report.		X
Basic authentication using username and password	X	
Extensive authentication using other methods verifying the user's identity, i.e. using one time codes.		X
Extensive logging of what the user does within the system.	X	
Every operation in the system should use transactions so no data are lost	X	

The main factors that need to be taken in to account when designing and building the system

Whilst an online-based bookkeeping system must be rigid to secure that the inputs act in accordance with the rules and laws of bookkeeping (Riksdagen 2007), it must also be customizable to ensure that enterprises can adapt the nominal ledger and the general journal as it desires. This implies that the system should be built in a way that allows user specific adjustments and expandability, taking the following factors into account:

- Inaccurate transaction data and/or erroneous numbers could have fatal implications to the user. This would not only generate user frustration, it could also result in costly lawsuits that would hurt the reputation of the system.
- A fundamental idea behind the design of the system is its high degree of accessibility. The possibility for a user to reach his/her bookkeeping from anywhere requires: (1) that users are able to login to the system using a typical web browser along with his/her username and password; (2) that the system is able to handle electronic traffic from computers at various geographic locations; and (3) that no unorthodox support applications need to be download to use the system.
- Storing transaction information in the system would do no good unless it could be made explicit in several ways. The system must be able to export information to printers and other applications such as Excel. An optional solution could be to allow the creation of sub-accounts with limited (observer) functionality.
- The information a user enters into the system contains financial enterprise data and should thus be regarded as confidential. It will consequently be necessary to store and cipher sensitive data and handle it with a high grade of security.
- Another important issue facing an online bookkeeping system is the two topics of speed and reliability. Users are unlikely to use the bookkeeping system unless it's as quick and reliable as other bookkeeping software. The issue of speed implicates that the system is built to support short loading times, that it's designed without unnecessary steps and that the bookkeeping is easy to navigate. The issue of reliability means that users should be given no reason to doubt the uptime of the bookkeeping system. Neither connectivity problems nor application breakdowns can be accepted.
- As different users may desire different bookkeeping methods, the online bookkeeping system should support several account plans – ex. BAS (BAS 2007a) – used in double entry bookkeeping.
- It must be possible to sort vouchers and alter whether data is shown alphabetically or chronologically.
- To avoid the handling of frustrating mistakes, users should have the ability to commit or reject the changes they've made. A useful but not necessary addition to the system would also be a history tracking tool that allowed users to make leaps in time by undoing (or redoing) changes.

Technologies and Risks

Developing web-based applications can be very time consuming as of the large number of technologies involved. For example, as a professional web-based developer you are required to master at least:

- HTML – Hyper Text Markup Language, a technique for displaying user interface over the Internet.
- CSS – Cascading Style Sheets, language that is commonly used to describe the presentation of a document written in HTML.
- JavaScript – Scripting language used to enhance user experience by allowing code to be executed on the client side of the application, the browser.

- A programming language – To allow users to interact with a webpage a programming language is used on the server side to handle logic how a page should be displayed and maybe connect to a database to fetch information.
- SQL – Structured Query Language, user for creating, updating, deleting and selecting data from a database.

By using a web-application framework, some of the technologies mentioned above can be handled by the framework. This simplifies the development of new web-based application when for example SQL are handled and generated by the framework. Most web-application frameworks will also provide security features and other convenient helper functions.

In this project we are planning on using the Ruby on Rails web-application framework. Ruby on Rails will provide us with many features like:

- Object-relational mapping.
- Application skeleton – a pre decided folder and file structure.
- A clean Model-View-Controller architecture.
- Easy to use programming language Ruby.
- Possibility to use pre-packaged functionality. For example a plug-in called acts_as_authenticated may be used for authentication of users.

Whilst there are many positive features using Ruby on Rails, there exist some risks:

- No distributed transactions – needed for making transactions over multiple databases. This will not be a problem at first but may cause trouble in the future if the system is required to modify an external source.
- Sometimes hard to find references and help with problems. Much of the Rails-knowledge on the Internet is found in badly indexed blogs. This will have an impact on productivity.
- Slow – for a large user base Ruby on Rails will not be very efficient because of the relatively small number of requests per second handled by a mongrel web server. This can be solved using multiple mongrel instances but with an overhead for each server.

We are also going to use some supporting non-critical business logic technologies to Ruby on Rails:

- HTML - de facto standard for web-based user interfaces
- CSS - de facto standard for styling web-based user interfaces
- JavaScript? - enhancement of user interface with client-side scripting

Glossary

- BAS – The BAS accounting plan is used by approximately 95% of all Swedish companies and could thus be seen as an unofficial standard template for the art of bookkeeping in Sweden (BAS 2007b).
- Accounting plan - Set of accounts used in double entry bookkeeping

User requirements

Non-functional

Unlike the functional requirements used to specify the exact behavior of Account This!, the non-functional requirements have first and foremost been mentioned to better depict the constraints and criteria that can be used to judge the general functionality of the system (Sommerville 2007, pp. 119). The non-functional – whether related to the product, the organization or fully external – are not directly linked with any individual system features but rather with the system itself (ibid., pp. 121-123). From the project description, it's evident that the intended customers (mainly consisting of people working in small firms) will desire security and reliability above anything else. Account This! must also come with a high degree of usability, making the program intuitive and easy to use (using the criteria seen below). A fourth major aspect category is standards. These categories alone however won't sell the product; several other categories of non-functional requirements are relevant as well.

Security

1. The Account This! system shall be able to resist 95 % of any malicious break-in attempts.
2. System administrators shall be able to make daily backups of user data to protect its users from any loss of information.
3. User accounts and user information shall be ciphered and unavailable to all except to the users themselves and to the system administrators.
4. As it's impossible to solve actions that have yet to be detected, all system failures must be logged by the Account this! system.

Usability

1. The layout of the user interface, such as the general journal and nominal ledger, must be customizable to better fit the variety of customer needs.
2. Users shall be able to grasp the general idea of the system in less than 3 minutes.
3. Users shall be able to learn how to use the system's functionality in less than 15 minutes, given that they have some previous experience of bookkeeping.
4. The structure of the website encompassing Account This! must be intuitive to all who've visited at least 100 websites.
5. The rate of button clicks leading to a page the user did not intend to see must be less than 3 %.
6. No unnecessary pages or boxes shall hinder the efficient use of Account This!. Each function shall consist of no more steps than would have been necessary with traditional pencil and paper bookkeeping.
7. A hyperlink to a helpful webpage containing sentences with explanatory text shall be accessible at all time.
8. Each user action shall be logged and stored by the system, granting users the opportunity to view a history of his/her past events.

Reliability

1. System failures caused by functions behaving different than they're supposed to must be at a failure rate of no more than 2 %.

2. Account This! shall be able to explain at least 95 % of the errors that have been detected (by showing a brief text message).
3. The probability of data corruption on failure must be less than 1 %.

Standards

1. All functions embedded in the bookkeeping system must meet the requirements imposed by Swedish law.
2. Account This! shall support all elements of the BAS accounting plan.
3. Each user must be able to change, save and use whatever standardized date format he or she prefers. Information concerning the chosen format must always appear in connection to any listed dates.
4. Each company must be able to change, save and use any existing currency format they desire. Information concerning the chosen format must always appear in connection to any listed value.
5. All corporate trademarks mentioned in the Account This! system must be written in accordance with Swedish laws and regulations.

Availability

1. The probability rate of function unavailability must be less than 0.5 % of the total system uptime.
2. The probability rate of webpage unavailability must be less than 1 % of the total system uptime.
3. Account This! shall be available from any Swedish ISP connection.
4. The system must be able to use from all computers equipped with modern web-browsers. No additional software shall be required.

Performance

1. The system must be built to ensure that users are provided some kind of response or feedback in less than one second from the related action.
2. The non-bandwidth consuming user/event response time must be imperceptible.

Scalability/extensibility

1. The time or effort required to implement changes in the software shall consist of no more than the energy (measured in time) that was put in to build the modification.
2. The Account this! code shall be provided with documentation enough to enable system administrators and maintenance personnel alike to grasp the purpose of the included methods.

Quality

1. All values shown in the system must be 99.9 % accurate to the numbers once entered.

Compability

1. The system shall support all operating systems that are compatible for, and equipped with, any of the modern web-browsers the system is designed for: Firefox 2.0, Internet Explorer 7 and Safari 3.

Functional

Users

1. The user shall be able to register a new company in the system providing information about the company and the initial user.
    1. The required information about the company is
        1. Name
        2. Company Form
        3. Organization Number
        4. Address (Standard Swedish address)
    2. The required information about the initial user is
        1. Name
        2. Address (Standard Swedish address)
        3. Phone-number
        4. Email
2. The user shall be able to login using their username and password.
3. The user shall be able to logout.
4. The user shall be able to update their information.

Companies

1. Each user in a company shall be able to easily create a new user in the company and then the system notifies the login information to the new user.
2. Each user in a company shall be able to update the companies' information.

Fiscal years

1. The user shall be able to create fiscal year.

Vouchers

1. The user shall be able to create vouchers
2. The user shall only be able see and edit vouchers belonging to his or hers company.
3. The users shall be able to create replacement vouchers to already existing vouchers it the voucher belongs to the current fiscal year. A replacement voucher will then act as the new voucher.
4. The users shall be able to strike out vouchers belonging to the current fiscal year. They will not be removed, but marked as removed.
5. To each non saved voucher belonging to the current fiscal year, the users shall be able to create/update/remove vouchers rows.
6. To each saved voucher belonging to the current fiscal year, the users shall be able to remove vouchers rows.

Support

1. The user shall be able to send questions in to support staff.
2. The user shall only be required to input the question itself.

Accounting plans

1. The user shall be able to create new accounting plans.

2. The user shall be able to create accounting plans from existing accounting plans (both accounting plans created by the user, and accounting plan templates provided by the system)
3. The user shall only be able to edit accounting plans used in fiscal years associated with the user's companies.
4. The user shall be able to remove accounting plans not used in a fiscal year.
5. The user shall be able to add accounts to an accounting plan.
6. The user shall be able to edit accounts if not used in any voucher.
7. The user shall be able to remove accounts from accounting plans if not used in any voucher.

Use cases

Users

Register new company

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.
- Oversight Tax Agency (the Skatteverket): Ensure that booking rules are obeyed.

Preconditions
The user is not authenticated.

Success Guarantee
The user has successfully registered a new company with a unique name.

Minimal Guarantee
The user has not registered a new company with the same name as a company that already exists in the system.

Main Scenario
1. The user is presented with a form for the company and the initial user.
2. The user fills out the form and enters all the required information and presses the register button.
3. If all the requested information is present the user is presented with confirmation screen that asks if all the entered information is correct. On this page there is two links. One that confirms the user’s information and registers the company and the user. And one that takes the user back to the form to change some of the information. The user clicks the register link.
4. The system registers the company, the initial user and creates a new fiscal year that belongs to the company. That fiscal year is set to the current fiscal year. A mail is mailed to the initial user with their login information.

Alternative Scenarios
From 2:

1. If some of the information is missing the user sees the form again with information about which information that are missing or which information that has not meet the validation (like an email address that is not an email address, or that the email the user tries to use is already in the system, etc..).
2. The user corrects his mistakes and presses register again. If the validation passes he will be in the main scenarios step nr 3.

User login

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.
- Oversight Tax Agency (the Skatteverket): Ensure that booking rules are obeyed.

Preconditions
The user is not authenticated.

Success Guarantee
The user is authenticated and aware of it.

Minimal Guarantee
The user is either not logged in or logged in as the user.

Main scenario
1. The user is presented with a login screen asking the user to fill out her username and password.
2. The user fills out the required credentials and presses the login button or the enter button.
3. The system validates the username together with the password against the user-database.
4. If the system considers the user valid it forwards the user to the start page inside the application.

Alternative Scenarios:
From 3:

1. If the system considers that the username or password is incorrect the user is redirected back to the login page with an error message that tells the user such.

User logout

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.
- Oversight Tax Agency (the Skatteverket): Ensure that booking rules are obeyed.

Preconditions
The user is authenticated.

Success Guarantee
The user is not authenticated and aware of it.

Minimal Guarantee
The user is either not logged in.

Main scenario
1. The user always has the ability to logout using the logout link.
2. If the user wants to logout he just clicks the link.
3. When the link is clicked the system cleans out the way the system keep track of that the user is logged in (i.e. a session variable or a cookie).
4. The system forwards the user to the main page with a message about that the user is now logged out.
5. The user reads the message about that he is now logged out.

Update user information

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.
- Oversight Tax Agency (the Skatteverket): Ensure that booking rules are obeyed.

Preconditions
The user is authenticated.

Success Guarantee
The user is authenticated and aware of it.

Minimal Guarantee
The user is either not logged in or logged in as the user.

Main Scenario
1. The user navigates to the page that shows his information, it's a link on this page that will take the user to the page where he could edit his information.
2. If the user clicks this link the system verifies that the user is actually the user that the information belongs to. If so the user is forwarded to the page where he can update his information.
3. The user updates his information as he wants and when he is finished he clicks a button to save the changes.
4. The system once again verifies that the user actually is the user that the information belongs and validates the new information just like the information was validated upon registration. If the user is the required used and the information passes validation the information is

saved and the user is forwarded to the pages that shows the updated information with a message that tells the user so.

Alternative Scenarios

From 2:

1. If the verification of the user fails but the user is an admin user he will be able to proceed and edit the user’s information anyway. This is valid for step 4 in the main scenario as well. The admin user should be able to save and update every user’s information in the system.

From 2:

1. If the user isn't the user that the information belongs to he will be presented a page that tells him so.

From 3:

1. If the users tries to hack the system and submits data in a malicious way the user will be presented a page that this is now allowed.

Companies

Create a new user in an company

Primary Actor

Customer/User

Stakeholders

- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.
- Oversight Tax Agency (the Skatteverket): Ensure that booking rules are obeyed.

Preconditions

The user is authenticated.

Success Guarantee

The user has registered a new user and the new user has been notified of such.

Minimal Guarantee

That the newly registered user is registered in the correct company.

Main Scenario

1. The user navigates to its company page. On this page there is a link to another page where the user can add new users to its company. The user presses this link and the system redirects the user to this page.
2. On this page the users fills out an email to the new user. When this is done the user clicks the link to add the new user.
3. When the link is clicked the system validates that the email address actually is an email address. If the validation passes the system adds a temporary user and sends a mail to the

new user asking him to fill out the rest of his information. Providing him with an URL to the page where he can do so.
4. The new user gets the mail and clicks the link.
5. The system forwards the user to the page where he can fill out the rest of the information required for the system. When the new user is done he clicks save.
6. Now the system validates his information and if the validates passes the user is added to the system under the right company.

Alternative Scenarios

From 3:

1. If the validation fails the user is redirected back to the page where he fills out the email address with a message that tells him that the validation failed.

From 6:

1. If the validation fails the new user is redirected back to the page where he fills out the information about himself with a message that tells him that the validation has failed.

Update the company information

Primary Actor

Customer/User

Stakeholders

- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.
- Oversight Tax Agency (the Skatteverket): Ensure that booking rules are obeyed.

Preconditions

The user is authenticated.

Success Guarantee

That the user has successfully updated the company's information.

Minimal Guarantee

That the information is updated for the correct company.

Main Scenario

1. The user navigates to the page that shows the company information, it's a link on this page that will take the user to the page where he could edit the company information.
2. If the user clicks this link the system verifies that the user is actually a user that has access to the company information. If so the user is forwarded to the page where he can update the company information.
3. The user updates the information as he wants and when he is finished he clicks a button to save the changes.
4. The system once again verifies that the user actually is a user that has access to the company and validates the new information just like the information was validated upon registration. If the user is the required user and the information passes validation the information is saved

and the user is forwarded to the pages that show the updated information with a message that tells the user so.

Alternative Scenarios

From 2:

1. If the verification of the user fails but the user is an admin user he will be able to proceed and edit the company information anyway. This is valid for step 4 in the main scenario as well. The administrator should be able to save and update every company information in the system.

From 2:

1. If the user doesn't have access to the company he will be presented a page that tells him so.

From 3:

1. If the users tries to hack the system and submits data in a malicious way the user will be presented a page that this is now allowed.

Fiscal year

Change fiscal year

Primary Actor

Customer/User

Stakeholders

- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions

The user is authenticated and therefore the user has a company.

Success Guarantee

The old fiscal year is not current and the new fiscal year is the current fiscal year.

Minimal Guarantee

The system logs how far it got, and the user is notified about the status of the operation.

Main scenario

2. The user navigates to the fiscal year page.
3. The user clicks on the link 'Start new fiscal year'
4. A confirmation telling the user that this will make all vouchers and
5. voucher rows from current fiscal year uneditable shows up.
6. The user clicks 'Confirm'.
7. The system locks all vouchers belonging to the current fiscal year and marks the new fiscal year as current.

Alternative scenario

From 2:

1. The user clicks 'Abort'
2. The system doesn't make any changes to the current fiscal year.

Vouchers

All actions that change the state of an earlier created voucher is only allowed if the voucher belongs to the current fiscal year.

Create a new voucher

Primary Actor

Customer/User

Stakeholders

- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions

The user is authenticated and therefore the user has a company.

Success Guarantee

A new voucher is created and saved.

Minimal Guarantee

The system logs how far it got, and the user is notified about the status of the operation.

Main scenario

1. The user navigates to the vouchers page.
2. The user selects 'New voucher'.
3. The user sees a view where new voucher rows can be added.
4. Selects **Create a new voucher row** multiple times.
5. The system saves the voucher and set the vouchers fiscal year to the company's current fiscal year.
6. The user returns to the vouchers page.

Alternative scenarios

From 5:

1. If the voucher contains zero rows, the user will be warned and the voucher will not be saved.

Remove a saved voucher

Primary Actor

Customer/User

Stakeholders

- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions

The user is authenticated and therefore the user has a company.

Success Guarantee
The voucher the user selected for removal is marked as removed.

Minimal Guarantee
The system logs how far it got, and the user is notified about the status of the operation.

Main scenario
1. The user navigates to the vouchers page.
2. The user clicks on the link named 'Mark as removed' beside the voucher in question.
3. The selected voucher is marked as removed.

Edit a saved voucher

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions
The user is authenticated and therefore the user has a company.

Success Guarantee
The voucher the user selected is overridden by a new one with the updated information.

Minimal Guarantee
The system logs how far it got, and the user is notified about the status of the operation.

Main scenario
1. The user navigates to the vouchers page.
2. The user clicks on the link named 'Create replacement voucher'.
3. The selected voucher is marked as removed.
4. A new voucher is created which is linked to the old one, but marked as a replacement for the old one.
5. The scenario continues as from step 3. in the **Create a new voucher** main scenario.

Voucher rows
All changes that apply to earlier saved voucher rows require that the voucher rows voucher belongs to the current fiscal year.

Create a new voucher row

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions
The user is authenticated and therefore the user has a company.

Success Guarantee
A new voucher row is created and saved.

Minimal Guarantee
The system logs how far it got, and the user is notified about the status of the operation.

Main scenario
1. The user selects 'New voucher row'.
2. The user fills in the sum and chooses an account for this row.
3. The user chooses weather it is a debit or credit row.
4. The user saves the row.

Alternative scenarios
From 4:

1. If the voucher row doesn't have an account or the sum is 0 the user will be warned and the voucher row will not be saved.

Edit a saved voucher row

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions
The user is authenticated and therefore the user has a company.

Success Guarantee
A new voucher row, which overrides the one that the user wished to edit, is created and saved.

Minimal Guarantee
The system logs how far it got, and the user is notified about the status of the operation.

Main scenario
1. The user navigates to the vouchers page.
2. The user clicks the link 'Edit voucher'.
3. The user is presented with all the voucher rows that belong to the selected voucher.
4. Each voucher row has a link with the text 'Replace' next to it.
5. The user clicks 'Replace' next to one of the voucher rows.
6. The selected voucher row is marked as removed.
7. A new voucher row appears marked as a replacement to the old.
8. The scenario continues as from step 2. in the **Create voucher row** main scenario.

Remove a saved voucher row

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions
The user is authenticated and therefore the user has a company.

Success Guarantee
The voucher row that the user selected is marked as removed.

Minimal Guarantee
The system logs how far it got, and the user is notified about the status of the operation.

Main scenario
1. The user navigates to the vouchers page.
2. The user clicks the link 'Edit voucher'.
3. The user is presented with all the voucher rows that belong to the selected voucher.
4. Each voucher row has a link with the text 'Mark as removed' next to it.
5. The user clicks 'Mark as removed' next to one of the voucher rows.
6. The selected voucher row is marked as removed.

Accounting plan

Create accounting plan

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions
The user is authenticated.

Success Guarantee
An accounting plan has been created with a name and associated with the user's company.

Minimal Guarantee
The accounting plan is not created and the user is aware of it.

Main scenario
1. The user navigates to the accounting plan page.
2. The user selects ‘Create accounting plan’.
3. The user is presented with a form

4. The user enters the name of the accounting plan in the text box.
5. The user presses the submit button.
6. The system creates a new accounting plan.
7. The system redirects the user to the accounting plan edit page.

Alternative scenario

From 5:

1. If the text box is empty, the user will be warned and the accounting plan will not be saved.

Duplicating an accounting plan

Primary Actor

Customer/User

Stakeholders

- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions

The user is authenticated.

Success Guarantee

The complete accounting plan has been duplicated and associated with the user's company.

Minimal Guarantee

The accounting plan is not duplicated and the user is aware of it.

Main scenario

1. The user navigates to the accounting plan page.
2. The user selects an accounting plan and then clicks ‘Duplicate accounting plan’.
3. The user is presented the create form.
4. The user enters the new name of the accounting plan.
5. The user chooses to which company the accounting plan will be copied.
6. The user press the submit button.
7. The system copies the accounting plan.
8. The system redirects the user to the new accounting plan edit page.

Alternative scenario

From 6:

1. If the text box is empty, the user will be warned and the accounting plan will not be copied.

From 6:

1. If the there already exists an accounting plan with the named specified in the text box for the company chosen in the selection drop-down, the user will be warned and the accounting plan will not be copied.

Deleting an accounting plan

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions
The user is authenticated.

Success Guarantee
The accounting plan has not been used in any fiscal year. The accounting plan has been deleted.

Minimal Guarantee
No other accounting plan than the user has selected is deleted.

Main scenario
1. The user navigates to the accounting plan page.
2. The user selects an accounting plan and then clicks 'Delete accounting plan'.
3. The user is presented with a confirmation prompt that asks if the user is sure about deleting the accounting plan.
4. The user chooses 'Yes'
5. The system removes the accounting plan.
6. The system redirects the user to the accounting plan page

Alternative scenario
From 3:

1. If the accounting plan is in use in any fiscal year, the system will deny the user from deleting the accounting plan.

From 4:

1. If the user has chosen 'No' in the confirmation prompt the accounting plan will not be deleted.

Adding accounts to an accounting plan

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions
The user is authenticated. An accounting plan exists.

Success Guarantee
A new, not already existing account, has been added to the accounting plan.

Minimal Guarantee
The system logs how far it got, and the user is notified about the status of the operation.

Main scenario
1. The user navigates to the accounting plan page.
2. The user selects an accounting plan and then selects 'Edit'.
3. The user selects 'New account'
4. The user is presented with a form for the new account.
5. The user fills out the form.
6. The user press the submit button.
7. The system saves the new account to the accounting plan.

Alternative scenario
From 6:

1. If an account with the same account number already exists in the accounting plan the user is warned and must choose another account number.

From 6:

1. If one of the required fields is not filled in the system warns the user and the new account will not be saved.

Removing accounts from an accounting plan

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions
The user is authenticated. An accounting plan exists.

Success Guarantee
The account is successfully removed from the chosen accounting plan.

Minimal Guarantee
The system logs how far it got, and the user is notified about the status of the operation.

Main scenario
1. The user navigates to the accounting plan page.
2. The user selects an accounting plan and then selects 'Edit'.
3. The user selects an account and then selects 'Delete account'
4. The user is presented with a prompt asking if the user is sure about deleting the account.
5. The user selects 'Yes'

6. The system removes the account from the accounting plan.

Alternative scenario
From 5:

1. If the user selects ‘No’, the account will not be removed.

From 5:

1. If the account to be removed is in use in any voucher the user will be denied deleting the account.

Support

Ask a question

Primary Actor
Customer/User

Stakeholders
- Customer/User: Wants accurate, fast entry, and no data corruption.
- Company: Wants profit and satisfied customers.

Preconditions
The user is authenticated.

Success Guarantee
The user's question is successfully submitted to the support system.

Minimal Guarantee
The system logs how far it got, and the user is notified about the status of the operation.

Main scenario
1. The user selects 'Help' from the menu.
2. The user is presented with the support form.
3. The user inputs her question.
4. The user submits the question.

Alternative scenarios
From 4:

1. If the text box is empty, the user will be prompted to write a question, otherwise the form will not be submitted.

Product standards
It’s of dire importance that Account this! adheres to the generally accepted accounting principles used in Sweden (Bokföringsnämnden 2007a). These principles are supported by a legal context mainly consisting of the Annual Accounts Act of 1995 (based on European directives) and the

Bookkeeping Act of 1999 (Svensk författningssamling 2007). They regulate the practice and impose a standard that enables comparison between, and investigation of, financial companies.

The accounting principles will have the following implications on Account this! (Skatteverket 2007; Svensk författningssamling 2007). First, both a journal (books of first entry) and a ledger (books of accounts) must be used. Second, the system must (due to the standards) be able to use existing bookkeeping templates like the BAS accounting plan (BAS 2007b). Third, the system must behave as can be expected for a double-entry bookkeeping system; each transaction shall result in one account being credited and another being debited the exact same amount. Fourth, each voucher must contain:

- a date
- a customer
- an event (time)
- a specification
- an amount.
- a unique identification number
- the seller's registration number (due to taxes)
- the name and address of the buyer
- the name and address of the seller
- an amount and explanation of the goods or service
- price before taxes
- the tax rate
- the amount of taxes to be paid

This collection of rules and regulations will naturally have great impact on how Account this! shall be developed.

System requirements

Functional

Users

1. The system shall be able to have users.
2. The system shall store the user's passwords with an irreversible hash-algorithm.

Companies

1. The system shall be able to have companies.
2. The company could have several users connected to the company.

Fiscal years

1. The system shall be able to have fiscal years.
2. Each company must have at least one fiscal year.
3. A company should not be able to have more than one current fiscal year.
4. Once a fiscal year is closed, it should not be able to be reopened.

Vouchers

1. The system shall be able to have vouchers.
2. All actions made on vouchers should be date and time stamped.
3. Once a voucher is saved, a voucher should never be able to remove.
4. Each voucher row belongs to an account and a fiscal year.
5. Each voucher row has a sum that is greater or less than zero. If it is greater than zero the voucher row is a credit row, otherwise it is a debit row.

Accounting plans

1. The system shall be able to have accounting plans.
2. An accounting plan is created for and belongs to one and only one fiscal year.
3. An account belongs to one and only one accounting plan (the same account number can however be used in multiple accounting plans).

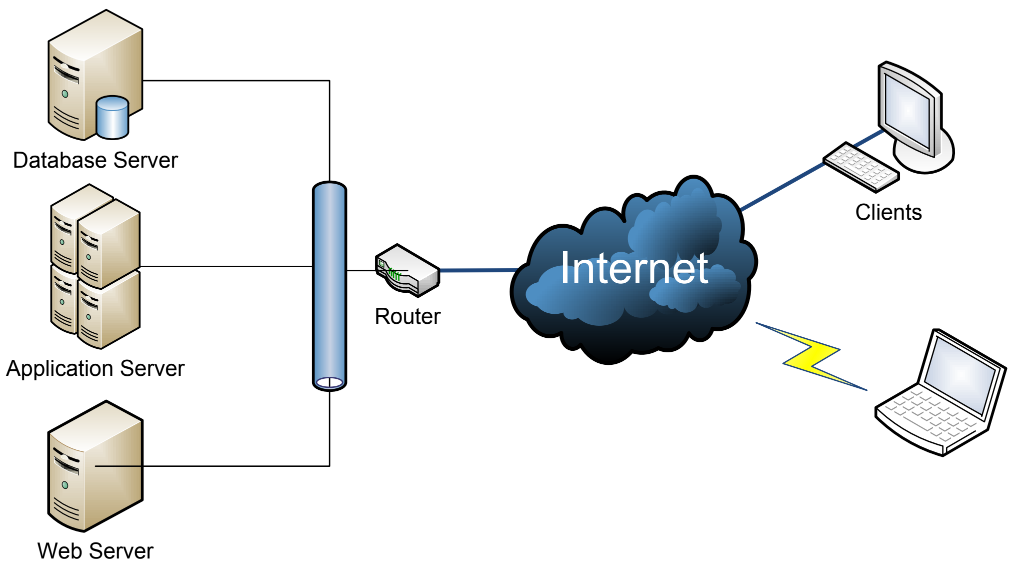
Support

1. Questions are sent as an e-mail to a configurable e-mail address.
2. Name and e-mail address to respond to shall be automatically, and transparent for the user, be provided from the user's profile.
3. A copy of the question shall be sent to the user's e-mail address.

System architecture

Overview

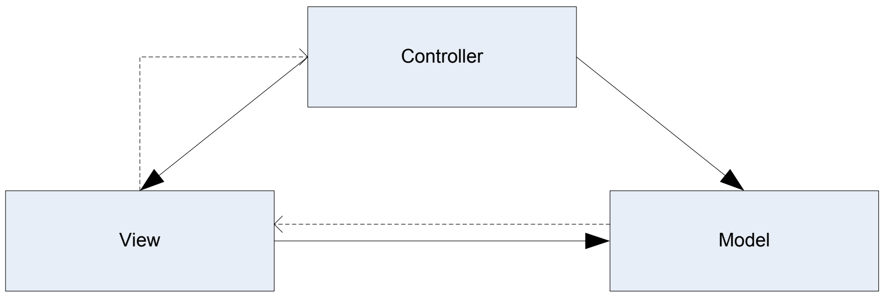
The overview of the anticipated server-client based system architecture.



All servers in the figure above are logical and can reside on one or more physical server as virtual servers.

Model-View-Controller architecture

The system is anticipated to be build with a Model-View-Controller pattern. Model View Controller is a design pattern that aims to modularize an application into 3 parts. The model represents the data for the application; the view represents the presentation; and the controller ties these two together and deals with user input.



System evolution

The dynamic edge of Account This! grants the system a good chance to remain efficient in a changing context. As Account This! is web based and requires no installations or fancy hardware, it should serve a good solution for firms unwilling to spend neither much time nor money on bookkeeping. Potential competitors could, in case they dare duplicate the functionality of Account This!, propose a threat to the popularity of our system and it's thus of great importance that Account This!, is made to be as expandable as is feasible. It must be possible to evolve the system by incorporating additional functionality, new options and helpful tools that our clients find attractive. Listening to the customers before customizing the system will thus be central for the continuous development and the strategic renewal of Account This!.

Appendices

Hardware

Client

Required hardware for the client is a standard personal computer with one of the three supported web browsers.

Server

Required hardware for the server is a standard computer able to run on the Ruby runtime.

References

BAS (2007a). Om BAS. Accessed on 25/11/2007, http://www.bas.se/

BAS (2007b). Kontoplan BAS 2008. Accessed on December 20, 2007:http://www.bas.se/kod/documents/BAS2008Kontoplan.pdf

Bokföringsnämnden (2007). BFN in English. Accessed on December 20, 2007: http://www.bfn.se/english.aspx

Riksdagen (2007). Svensk författningssamling (SFS). Accessed on 11/11/2007, http://www.riksdagen.se/webbnav/index.aspx?nid=3911&dok_id=SFS1999:1078&rm=1999&bet=1999:1078

Skatteverket (2007). Bokföring - vad kräver lagen? Tips m.m. Accessed on December 20, 2007: http://www.skatteverket.se/fordigsomar/arbetsgivareinfotxt/bokforing.4.18e1b10334ebe8bc80005195.html

Sommerville, I. (2007). Software Engineering, Eight Edition, Pearson Education, Essex, England.

Svensk författningssamling (2007). Bokföringslag (1999:1078). Accessed on December 20, 2007: http://www.riksdagen.se/webbnav/index.aspx?nid=3911&bet=1999:1078

PrimitiveType Web Development Glossary. Accessed on January 4, 2008: http://www.primitivetype.com/glossary/mvc.php