APPENDIX

This appendix contains a more detailed description of the most important functions and variables within each class in the project.

CAction
cards_Deal()
Assembles information needed to make a deal call to the server. The information is then POST'ed to the server by the CHTTPRequest. Then response from the server is sent to objTableGFX.setButtons which updates the graphics according to the parameters contained within it
A similar function, to cards_Deal(), exists for all actions the user can take while playing such as hit, stand, double etc.
login()
Requests the login page and tries to log in with given username and password as input. Establishes the gaming session if successful.

CBJButton
Enable()/Disable()
Enables and disables a button. A disabled button is not drawn to the screen and doesn't trigger any events if the screen is touched in a position inside its area.
drawButton()
Draws the current button on the screen.
isClicked()
Checks whether a button has been pressed or not.

CBlackJackDoc
OnNewDocument()
Initializes the data structure for the document such as all the buttons.

CBlackJackView
OnDraw()
OnDraw is called by the framework to render an image of the document.
The function calls drawBackground() to render the background, CBJButton:drawButton() to draw all buttons and CTableGFX:drawText() to display draw text.
OnLButtonDown()
Every time the pen touches the screen OnLButtonDown is called.
The function then checks if the coordinates at which the screen was touched correspond to the area encompassed by any active buttons. If they do, the actions specific to that button are called.
updateButtons()
Draws button to the screen according to their buttonStatus. A buttons buttonStatus depends on the game state.

CConnection
Connect()
Connect creates the root HINTERNET handle (created and used by the included WinInet class functions) to a given server and port. The HINTERNET handle, which is

the programmatical equivalent of a successful connection to a server, is then used by all subsequent HTTP protocol functions.
Disconnect()
Closes the HTTP session and the HINTERNET handle.

CHTTPRequest
makeHTTPRequest()
This function is used to make HTTP requests, and to retrieve the replies they generate.

CLoginDlg
OnFullScreen()
Puts the dialog in full screen mode and hides the command and task bar in order to display as much as possible.
OnLButtonDown()
Every time the pen touches the screen OnLButtonDown is called.
The function then checks if the coordinates at which the screen was touched correspond to the area encompassed by any active buttons. If they do, the actions specific to that button are called.
For example, ff the login button is pressed the login function in CAction is called. If cancel is pressed the application returns to the initial splash screen.

CSession
Get and set functions for the current member variables:

- *BetSize* (how big the stake is for the current)
- *BV_EngineId* (needed in order to maintain the session at the server)
- *BV_SessionId* (needed in order to maintain the session at the server)
- *GameId* (id for current game round)
- *Dealer and player cards*
- *Port* (port, at the gaming server, used for communication)
- *Server* (IP address of the gaming server)
- *SSL* (specifies if SSL is to be used)
- *Saldo* (wallet account Blackjack)
- *PlayerId* (id of the player currently logged in)
- *PortSessionId* (BV Session ID for formatted for POST requests)
- *TableMax* (The maximum bet of the current table)
- *TableMin* (The minimum bet of the current table)
- *TableType* (The gaming table currently being played)

CTableGFX
drawCard()
Draws a single card on the destination CDC (Class Device Context) (In our case always the background playing area)
setButtons()
Parses the input data, sets most game specific variables (dealercards, playercards, wallet Blackjack etc.) and decides which buttons are to be shown to the user the next time the screen is refreshed.

Globals
An code example of how globals works in practice:

```
CSession objSession;
```

When a class wants to access the global class object it includes the class file and references the global object as an “extern” object.

```
#include "Session.h"
…
extern CSession objSession;
…
objSession.setPlayerCards1("");
```

Globals used:

• *objAction*	(game functionality)
• *objConnection*	(server connection)
• *objHTTPRequest*	(server communication)
• *objSession*	(session functionality)
• *objTableGFX*	(table graphics)
• *objUtils*	(utilities)

Login functionality

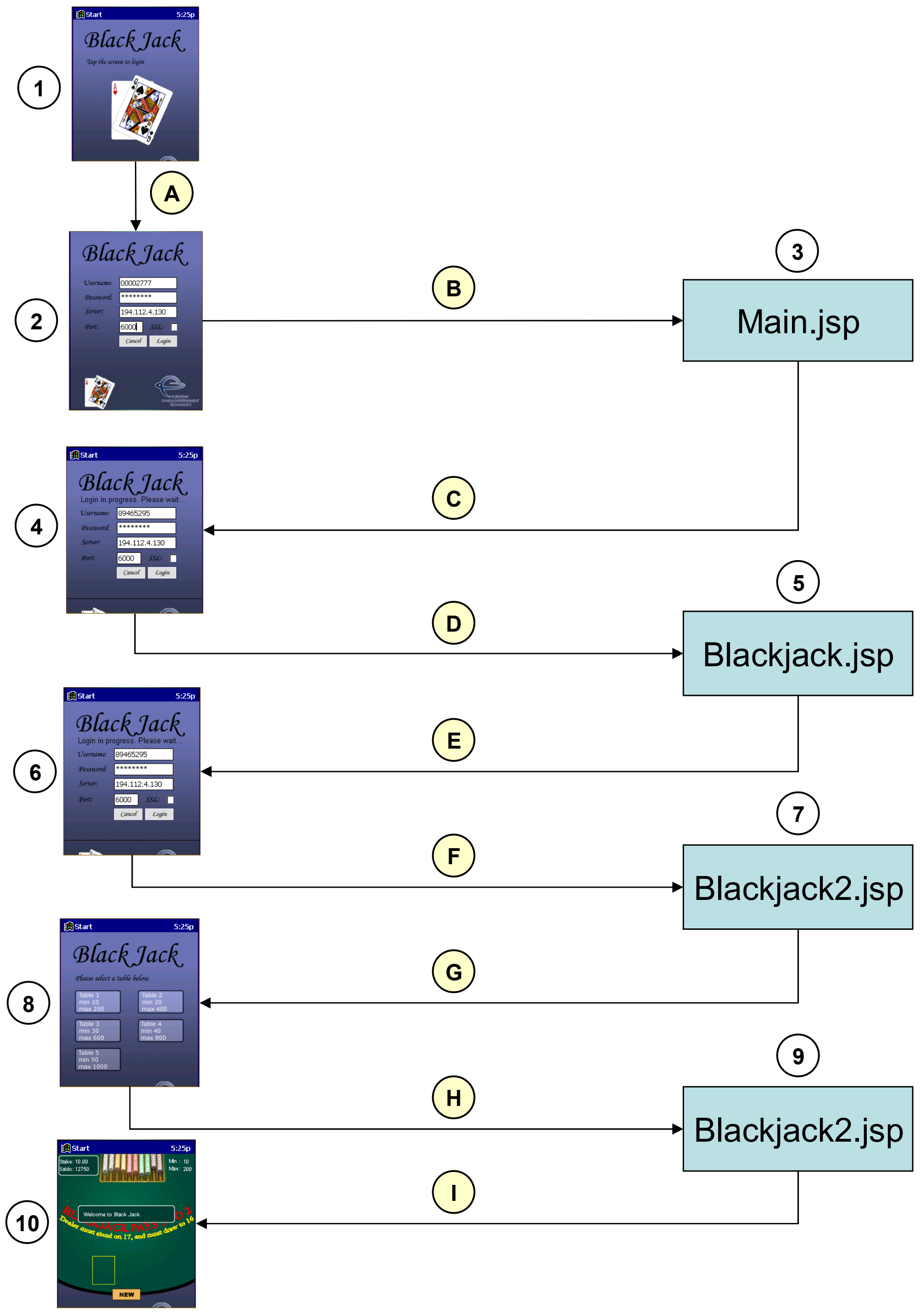


Figure 1- Overview of the requests and responses generated during the login process.

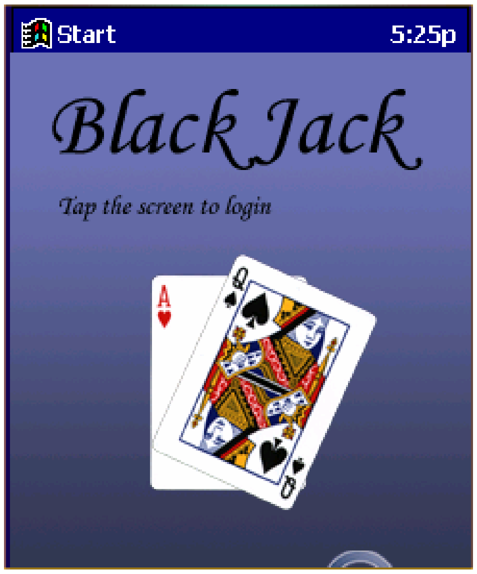
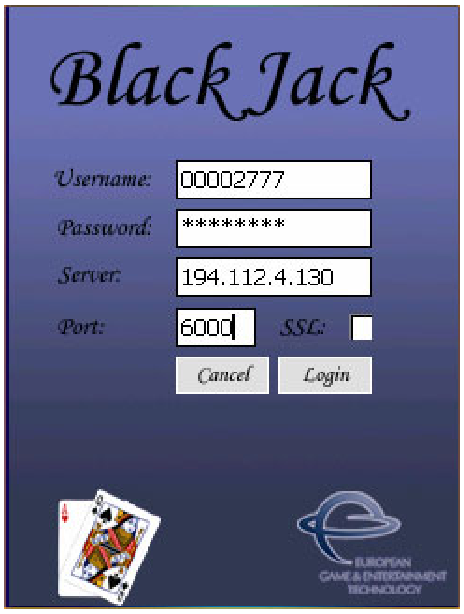


Figure 2 – Splash screen

1. The splash screen shown to the user at start-up of Blackjack.

A. The user taps the login screen.



Username: 00002777
Password: ********
Server: 194.112.4.130
Port: 6000
SSL: ☐

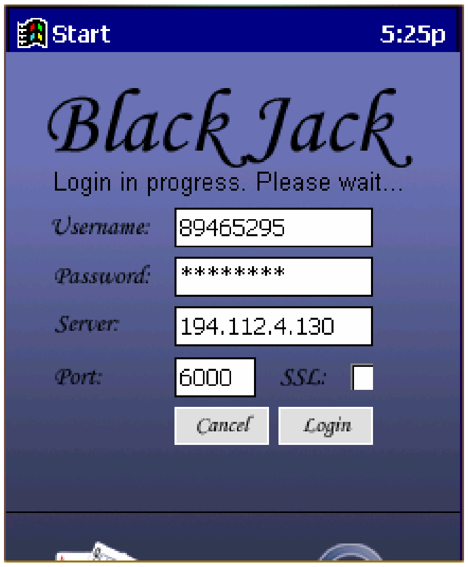
Figure 3 – Login screen

2. The user is shown the login screen. He is asked to fill in the following details:
   - **Username:** The 8 digit username he has been given upon registering.
   - **Password:** The 8 digit password he chose upon registering.
   - **Server:** The IP address of the server he wishes to log on to.
   - **Port:** The port of the server he wishes to log on to.
   - **SSL:** Checked if communication is to be conducted over an SSL-encrypted connection. Left unchecked otherwise.

B. The user taps the login button. The PDA establishes a socket-connection with the server specified in the form. The information he entered is submitted to main.jsp on the server. Unless something unforeseen happens, the connection is kept open for the duration of the session.

3. Main.jsp validates the submitted information.

C. The HyperText Mark-up Language (HTML) page generated by Main.jsp is sent to the client.



Login in progress. Please wait...

Username: 89465295

Password: ********

Server: 194.112.4.130

Port: 6000 SSL: ☐

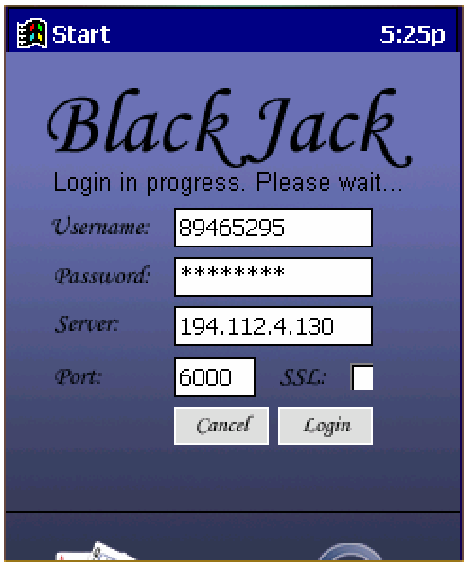
Figure 4 – Login screen, communicating with the server

4. The user is shown the "Login in progress" screen indicating that communication is going on with the server he wanted to log on to. The HTML page returned by Main.jsp (C) is parsed. The session unique identifiers, `BV_SessionId` and `BV_EngineId` are extracted and stored in memory on the PDA. These identifiers are needed in all communication within the same session to validate the requests.

D. The following HTTP GET request is made by the PDA:
GET
/gaming1/casino/Blackjack.jsp?BV_SessionID=@@@@0493346568.0994156215@@@@&BV_EngineID=callgcjfmhjbemhcffgcicfjf.0

5. BlackJack.jsp parses the request.

E. The HTML page generated by BlackJack.jsp is sent back to the client.



Login in progress. Please wait...

Username: 89465295

Password: ********

Server: 194.112.4.130

Port: 6000 SSL:

Figure 5 – Login screen, request parsed

6. The user is still shown the "login in progress" screen. The PDA parses the HTML page returned by BlackJack.jsp (E). Player_id is extracted and stored for the duration of the session. This information is needed later in other requests.

F. A HTTP POST request is to BlackJack2.jsp. The following information is POST'ed along with the page request:

```
Action=init
BV_SessionID=%40%40%40%400493346568.0994156215%40%40%40%40
BV_EngineID=callgcjfmhjbemhcffgcicfjf.0
Game_id=
Player_id=12000
Stake=
Table_type=1
```

Parameter explanation

Action	Action specifies which action the user has taken. Init is a request for game initialization.
BV_SessionID	Used to validate the request.
BV_EngineID	Used to validate the request.
Game_id	Of no importance in this request.
Player_id	Identifier within the system for the player. Read from the value stored in step 6.
Stake	Of no importance in this request.
Table_type	Of no importance in this request.

7. BlackJack2.jsp parses the request.

G. The HTML page generated by BlackJack.jsp is sent back to the client. The reply contains the following information:

```
Action = init
BV_SessionID = @@@@1654501732.0994319026@@@@
BV_EngineID = callgfcmllhbemhcffgcicfif.0
Player_id = 12000
Table1 = 10,200
```

```
Table2 = 20,400
Table3 = 30,600
Table4 = 40,800
Table5 = 50,1000
Soft_double_values = 19,20
Double_values = 9,10,11
Extra_X = 0
Extra_XxX = 0
0_games = 0
Table_type = 5
Demo_games = -1
Sfx = 1
Music = 1
Speech = 1
```

Parameter explanation

Action	Confirmation to the client of the action that has been taken. In this case init, which means that the serverside has initialized Blackjack for the player.
BV_SessionID	Used to validate the request.
BV_EngineID	Used to validate the request.
Player_id	Identifier within the system for the player. Read from the value stored in step 6.
Table1	The minimum and maximum values that a bet can take if the player chooses to play table 1 (explained later)
Table2	The minimum and maximum values that a bet can take if the player chooses to play table 2 (explained later)
Table3	The minimum and maximum values that a bet can take if the player chooses to play table 3 (explained later)
Table4	The minimum and maximum values that a bet can take if the player chooses to play table 4 (explained later)
Table5	The minimum and maximum values that a bet can take if the player chooses to play table 5 (explained later)
Soft_double_values	Parameters not used within the PDA version of
Double_values	Parameters not used within the PDA version of Blackjack
Extra_X	Parameters not used within the PDA version of Blackjack
Extra_XxX	Parameters not used within the PDA version of Blackjack
0_games	Parameters not used within the PDA version of Blackjack
Table_type	Parameters not used within the PDA version of Blackjack
Demo_games	Parameters not used within the PDA version of Blackjack
Sfx	Parameters not used within the PDA version of Blackjack
Music	Parameters not used within the PDA version of Blackjack
Speech	Parameters not used within the PDA version of Blackjack

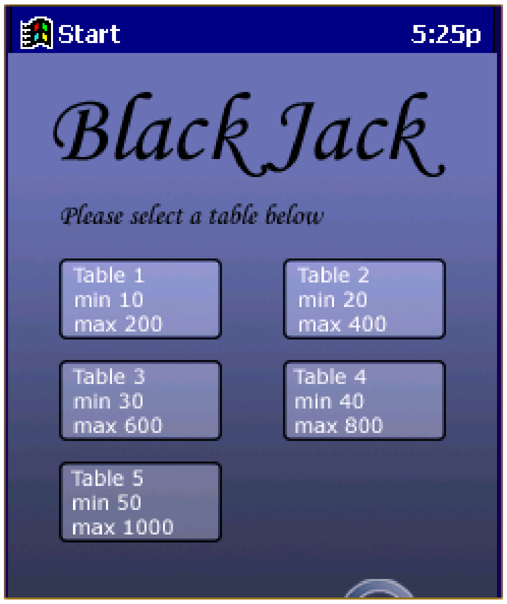


Figure 6 – Table selection screen

8. The user is shown a screen where he is allowed to choose which table he wants to play. The 5 available tables have different minimum and maximum bets. These limits are taken from the Table1, Table2, Table3, Table4 and Table5 parameters returned by the previous init request.

H. The user chooses one of the tables by tapping it. A HTTP POST request is to BlackJack2.jsp. The following information is POST'ed along with the page request:

```
Action=start
BV_SessionID=%40%40%40%401654501732.0994319026%40%40%40%40
BV_EngineID=callgfcmllhbemhcffgcicfif.0
Game_id=
Player_id=12000
Stake=0.00
Table_type=5
```

Parameter explanation

Action	Action specifies which action the user has taken. Start is a request for gaming to begin.
BV_SessionID	Used to validate the request.
BV_EngineID	Used to validate the request.
Game_id	Of no importance in this request.
Player_id	Identifier within the system for the player. Read from the value stored in step 6.
Stake	Of no importance in this request.
Table_type	Identifier for the table the user has chosen. Used to set the minimum and maximum wagers that will be allowed within the following game session.

9. BlackJack2.jsp parses the request.

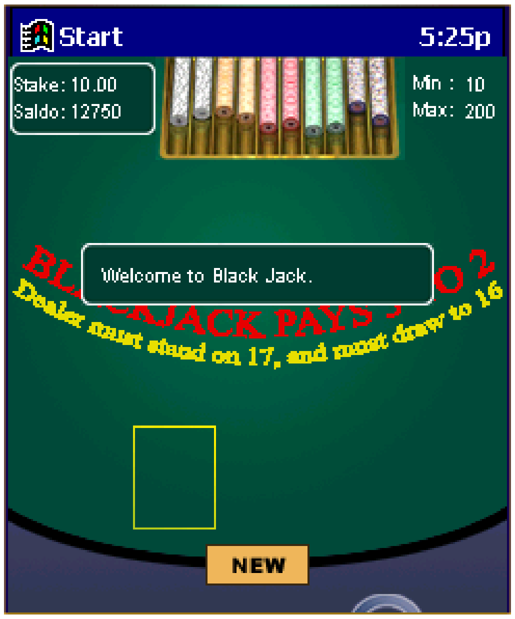
I. The HTML-page generated by BlackJack.jsp is sent back to the client. The reply contains the following information:

```
Action = start
BV_SessionID = @@@@1654501732.0994319026@@@@
BV_EngineID = callgfcmllhbemhcffgcicfif.0
Game_id = 1780822
Player_id = 10134
Hand_0_win = 0.00
Hand_1_win = 0.00
Dealercards =
Playercards1 =
Playercards2 =
State = 0
house_total =
player_0_total = 0
player_1_total = 0
saldo = 11955
stake = 0
insurance = 0
Table_type = 5
Min = 50
Max = 1000
Chip1 = 10
Chip2 = 20
Chip3 = 50
Chip4 = 100
Chip5 = 500
```

I.

Action	Confirmation to the client of the action that has been taken. In this case start, which means that the serverside has started Blackjack for the player and is ready to start playing.
BV_SessionID	Used to validate the request.
BV_EngineID	Used to validate the request.
Game_id	Unique identifier for this particular game. Every game within a session is given a unique Game_id. This is used at the serverside to be able to track individual games.
Player_id	Identifier within the system for the player.
Hand_0_win	Of no importance in this request.
Hand_1_win	Of no importance in this request.
Dealercards	Of no importance in this request.
Playercards1	Of no importance in this request.
Playercards2	Of no importance in this request.
State	Signifies which state Blackjack is in relation to the user that has logged in. Explained in detail in the section which details Blackjack gaming.
house_total	Of no importance in this request.
player_0_total	Of no importance in this request.
player_1_total	Of no importance in this request.
saldo	The saldo of the player. Stored by the PDA and used to determine

	how much the user can bet. Makes it harder for the user to make malformed requests.
stake	Of no importance in this request.
insurance	Of no importance in this request.
Table_type	Identifier for the table the user has chosen. Used to set the minimum and maximum wagers that will be allowed within the following game session.
Min	Parameters not used within the PDA version of Blackjack
Max	Parameters not used within the PDA version of Blackjack
Chip	Parameters not used within the PDA version of Blackjack
Chip2	Parameters not used within the PDA version of Blackjack
Chip3	Parameters not used within the PDA version of Blackjack
Chip4	Parameters not used within the PDA version of Blackjack
Chip5	Parameters not used within the PDA version of Blackjack



Welcome to Black Jack.

Figure 7 – User logged in successfully

10. The user is shown the "Welcome" interface for Blackjack. At this point all initialization needed by the system has been made, and the user can commence with normal gaming.

Playing Blackjack

The most complex and the largest part of the application is of course the logic required to actually play the game. The underlying diagram and text explains the different events associated with gaming. A more detailed and technical explanation is available in appendix. B**.**

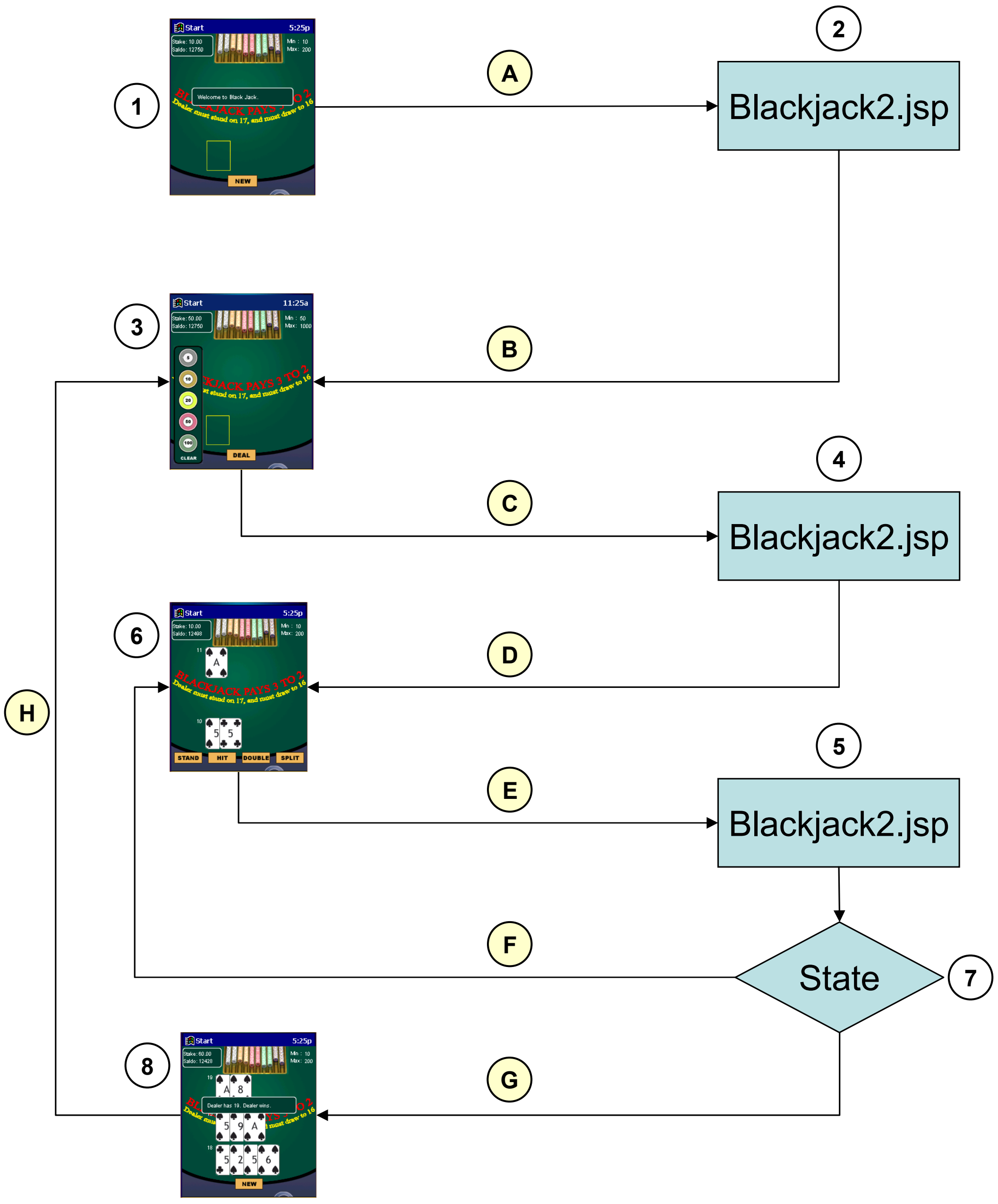


Figure 8 – Overview of the requests and responses that are generated during play of blackjack.

1. The user is shown the "Welcome" interface for Blackjack. Preceding this is the login procedure described earlier in this document.

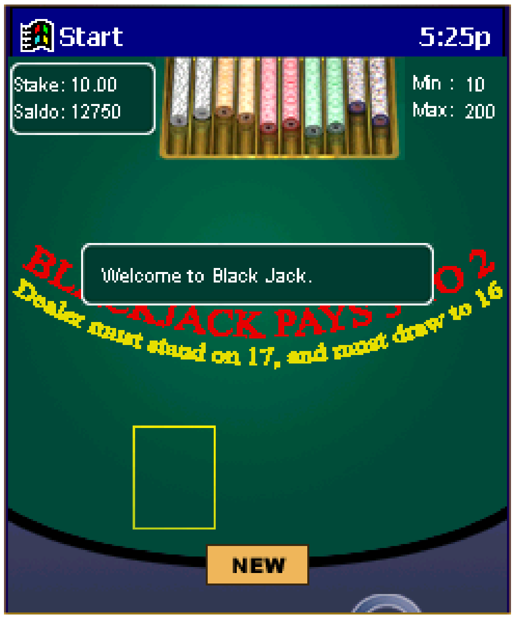


Figure 9 – Start a new game

A. The user chooses to initiate a new game by tapping the "NEW" button. A HTTP POST request is to BlackJack2.jsp. The following information is POST'ed along with the page request:

```
Action=new
BV_SessionID=%40%40%40%401654501732.0994319026%40%40%40%40
BV_EngineID=callgfcmllhbemhcffgcicfif.0
Game_id=1780822
Player_id=10134
Stake=0.00
Table_type=
```

Parameter explanation

Action	Action specifies which action the user has taken. `new` is a request for gaming to begin.
BV_SessionID	Used to validate the request.
BV_EngineID	Used to validate the request.
Game_id	Unique identifier for this particular game. Every game within a session is given a unique `Game_id`. This is used at the serverside to be able to track individual games.
Player_id	Identifier within the system for the player.
Stake	Of no importance in this request.
Table_type	Identifier for the table the user has chosen. Used to set the minimum and maximum wagers that will be allowed within the following game session.

2. BlackJack2.jsp parses the request and generates the reply.

B. The HTML-page generated by BlackJack.jsp is sent back to the client. The reply contains the following information:

```
Action = new
BV_SessionID = @@@@1654501732.0994319026@@@@
BV_EngineID = callgfcmllhbemhcffgcicfif.0
Game_id = 1780823
Player_id = 10134
Hand_0_win = 0.00
Hand_1_win = 0.00
Dealercards =
Playercards1 =
Playercards2 =
State = 0
house_total =
player_0_total = 0
player_1_total = 0
saldo = 11955
stake = 0
insurance = 0
```

Parameter explanation

Action	Confirmation to the client of the action that has been taken. In this case **new**, which means that the serverside is ready to deal the player his hand.
BV_SessionID	Used to validate the request.
BV_EngineID	Used to validate the request.
Player_id	Identifier within the system for the player. Read from the value stored in step 6.
Hand_0_win	Of no importance in this response.
Hand_1_win	Of no importance in this response.
Dealercards	Of no importance in this response.
Playercards1	Of no importance in this response.
Playercards2	Of no importance in this response.
State	Explained in detail further down.
house_total	Of no importance in this response.
player_0_total	Of no importance in this response.
player_1_total	Of no importance in this response.
saldo	The balance of the player. Stored by the PDA and used to determine how much the user can bet. Makes it harder for the user to make malformed requests.
stake	Of no importance in this response.
insurance	Of no importance in this response.

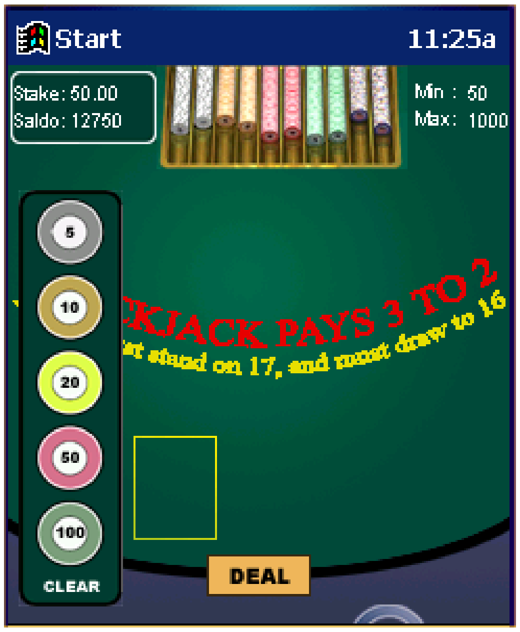


Figure 10 – Bet screen

3. The user is shown the screen where he can decide how much he wants to bet. The default value is the table minimum, shown top right. The player bets by tapping the markers. His current bet is shown top left. The application only allows him to make bets that are larger or equal to the table minimum, smaller or equal to the table maximum, and smaller or equal to his saldo.

C. The user instructs the server to deal his initial cards by tapping the "DEAL" button.. A HTTP POST request is to BlackJack2.jsp. The following information is POST'ed along with the page request:

```
Action=deal
&BV_SessionID=%40%40%40%401654501732.0994319026%40%40%40%40
&BV_EngineID=callgfcmllhbemhcffgcicfif.0
&Game_id=1780823
&Player_id=10134
&Stake=50.00
&Table_type=
```

Parameter explanation

(Only parameters whose importance within the request differ from A. are listed. All unlisted parameters serve the same purpose within this request as within A)

Action	Action specifies which action the user has taken. `deal` is a request for the server to deal the player and dealer their initial cards.
Stake	Tells the server how much the player has decided to bet on the hand he is about to receive.

4. BlackJack2.jsp parses the request and generates the reply.

D. The HTML-page generated by BlackJack.jsp is sent back to the client. The reply contains the following information:

```
Action = deal
BV_SessionID = @@@@1654501732.0994319026@@@@
BV_EngineID = callgfcmllhbemhcffgcicfif.0
Game_id = 1780823
Player_id = 10134
Hand_0_win = 0.00
Hand_1_win = 0.00
Dealercards = 1
Playercards1 = 3,8
Playercards2 =
State = 0
house_total = 11
player_0_total = 11
player_1_total = 0
saldo = 11905
stake = 50.00
insurance = 1
```

Parameter explanation
(Only parameters whose importance within the response differ from B. are listed. All unlisted parameters serve the same purpose within this response as within B)

Action	Confirmation to the client of the action that has been taken. In this case **`deal`**, which means that the player is receiving the first cards of his hand.
Dealercards	The cards that the dealer has received so far. Cards are taken from a pack of 4 complete decks. Each card has a unique number ranging from 1 to 208. The PDA receives a comma separated list with the card identifiers and shows these to the user.
Playercards1	The cards that the player has received so far for his first hand. Cards are taken from a pack of 4 complete decks. Each card has a unique number ranging from 1 to 208. The PDA receives a comma separated list with the card identifiers and shows these to the user.
Playercards2	The cards that the player has received so far for his second hand. Can only contain data if the player has **`split`**. Cards are taken from a pack of 4 complete decks. Each card has a unique number ranging from 1 to 208. The PDA receives a comma separated list with the card identifiers and shows these to the user.
State	Explained in detail further down.
house_total	The sum of the values of each of the cards that the dealer has received.
player_0_total	The sum of the values of each of the cards that the player has received for his first hand.
player_1_total	The sum of the values of each of the cards that the player

	has received for his second hand. Can only contain data if the player has split.
insurance	Indicates whether the insurance dialog should be shown to the user. I.e., the dealer may have Blackjack, and the user shall be given the option of taking insurance. This could have been deduced within the PDAs own logic based upon the dealt cards, previous designers have opted to make this easier by sending the information to the client. A 1 indicates that the dialog should be shown, anything else that it shouldn't.

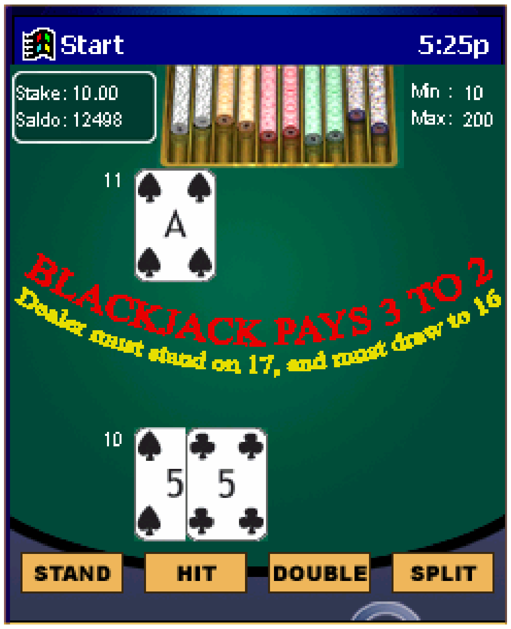


Figure 11 – Game screen

5. The following dialog is shown to the user. Depending on which cards the user has been dealt a different subset of buttons, representing the actions the user can take, is shown. In figure 11 the user has been dealt such a hand that all actions (`Stand, Hit, Double, and Split`) are valid. The user is also graphically shown the cards that have been dealt so far.

E. The user instructs the server to stand, hit, double or split by tapping the one of the buttons he is shown. A HTTP POST request is to BlackJack2.jsp. The following information is POST'ed along with the page request:

`Action=` one of [`stand,hit,double,split`]
&BV_SessionID=%40%40%40%401654501732.0994319026%40%40%40%40
&BV_EngineID=callgfcmllhbemhcffgcicfif.0
&Game_id=1780823
&Player_id=10134
&Stake=50.00
&Table_type=

Parameter explanation
(Only parameters whose importance within the request differ from A or C. are listed. All unlisted parameters serve the same purpose within this request as within A or C)

Action	Action specifies which action the user has taken. Action can be one of: `Stand`, The user doesn't want any more cards for the active hand. `Hit`, The user wants one more card for the active hand. `Double`, The user wants to double the stake for the current hand. `Split`, The user wants to split his hand into two separate hands. The user can only take actions that are legal in respect to the cards he has already been dealt. On the PDA this is made sure of by only showing the user buttons for legal actions.

6. BlackJack2.jsp parses the request and generates the reply.

7. As a part of step 5 the server generates a statecode. The statecode tells the application what the result of the users action is. The following states are possible:

   0 = The game has not yet been won by any part
   1 = Dealer wins with better cards
   2 = Dealer wins by Blackjack
   3 = Player wins with better cards
   4 = The player wins one hand, the dealer the other. After a split
   5 = The dealer busts
   6 = The player wins with Blackjack
   7 = Both hands are a draw after a split
   8 = The game is a draw
   9 = The player loses one hand and draws the other after a split
   10 = The player draws hand one and wins hand two after a split.
   11 = The player wins hand one and draws hand two after a split.
   12 = Both players have Blackjack, the game is a draw.
   26 = The game is closed
   50 = Not enough money to cover the action.
   55 = Erroneous request or server error.
   60 = Too big or too small stake.
   65 = The players personal gaming limit is exceeded

F. If the state is 0, i.e. no one has yet won the hand, the following response is sent to the client.:

   `Action =` one of [`stand,hit,double,split`]
   BV_SessionID = @@@@1654501732.0994319026@@@@
   BV_EngineID = callgfcmllhbemhcffgcicfif.0
   Game_id = 1780823
   Player_id = 10134
   Hand_0_win = 0.00
   Hand_1_win = 0.00
   Dealercards = 1

```
Playercards1 = 3,8
Playercards2 =
State = 0
house_total = 11
player_0_total = 11
player_1_total = 0
saldo = 11905
stake = 50.00
insurance = 1
```

Parameter explanation

The parameters are the same as explained in step E.

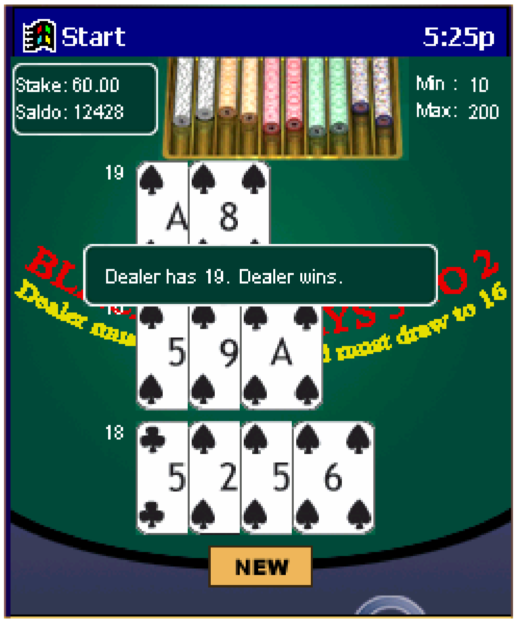
Because the game hasn't yet been won by either part, the game now returns to step 5.

G. If the state-code is something else than 0, i.e. the game has been won by someone, the following response is sent to the client:

```
Action = one of [stand,hit,double,split]
BV_SessionID = @@@@1654501732.0994319026@@@@
BV_EngineID = callgfcmllhbemhcffgcicfif.0
Game_id = 1780823
Player_id = 10134
Hand_0_win = 0.00
Hand_1_win = 0.00
Dealercards = 1,7
Playercards1 = 3,8,2,5
Playercards2 =
State = 8
house_total = 18
player_0_total = 18
player_1_total = 0
saldo = 11955
stake = 50.00
insurance = 0
```

Parameter explanation
(Only parameters whose importance within the response differ from D. are listed. All unlisted parameters serve the same purpose within this response as within D)

Hand_0_win	The amount of money, if any, the players first hand won him.
Hand_1_win	The amount of money, if any, the players second hand won him.



Dealer has 19. Dealer wins.

Figure 12 – Game finished

8. The player is shown a screen where he is informed of who won.

H. The player starts a new hand by pressing “New”. The request is the same as step B.

Parameters not used within the PDA version of Blackjack

Soft_double_values	Specifies the soft values, i.e. when the ace is counted as a 1 and not 11,which values the player is allowed to double on.
Double_values	Specifies the values which a player is allowed to double on.
Extra_X	Specifies how much, if at all, the user should receive as a bonus if he draws three 7s
Extra_XxX	Specifies how much, if at all, the user should receive as a bonus if he succeeds in drawing 5 cards or more without busting.
0_games	Used to tell the client that the user has an unfinished game which has to be played to an end before he can play a new hand.
Table_type	Specifies which table the user is playing on. Table is a definition which was invented in order to differentiate different intervals the user is allowed to bet within. For example one table might allow bets between 1 and 10 markers whereas another table allows bets within 10 and 100 markers. The parameter is of no importance to the server, it is there to give the client the notion of playing at a table with higher or lower stakes.
Demo_games	Specifies if the game being played is in demo mode. If in demo mode no real money can be won or lost.
Sfx	Specifies if sound effects should be switched on.
Music	Specifies if music should be switched on.
Speech	Specifies if speech should be switched on.
Min	The minimum bet for the current table.
Max	The maximum bet for the current table.
Chip	The value of the first chip.
Chip2	The value of the second chip.
Chip3	The value of the third chip.
Chip4	The value of the fourth chip.
Chip5	The value of the fifth chip.

TERMINOLOGY

Abbreviations

API	Application Programming Interface
BMP	Bitmap Graphics
BV	BroadVision
CAD	Computer Aided Design
CDC	Class Device Context
CE	Compact Edition
DB	Database
EGET	European Game & Entertainment Technology
EXE	Executable
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
IR	Infrared
JPG	Joint Photographic Group
JSP	Java Server Pages
Mb	Megabytes
MDI	Multiple Document Interface
MFC	Microsoft Foundation Classes
OS	Operating System
PC	Personal Computer
PDA	Personal Digital Assistant
SDI	Single Document Interface
SIP	Soft Input Panel
SSL	Secure Sockets Layer
URL	Uniform Resource Locator
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network

REFERENCES

Ian Sommerville (2000), *Software Engineering (6th Edition)*, Addison-Wesley

Background

Solutions overview,
http://www.eget.fi/solutions.html,
2002-06-26

The Evolution of the Pocket PC as a Business Tool,
http://www.microsoft.com/mobile/enterprise/papers/evol.asp,
2001-11-02

Blackjack Rules,
http://194.112.4.91/demo_gaming1/casino/instructions/info_bot.jsp?BV_SessionID=@
@@@1023148488.1029175850@@@@&BV_EngineID=cadcekhjgeglbemicffgcicog.
0&category=Spelregler,
2002-08-12

Technology

Microsoft Windows CE: An Overview,
http://www.wirelessdevnet.com/channels/pda/training/winceoverview.html,
2002-08-05

Microsoft eMbedded Visual Tools Product Information,
http://msdn.microsoft.com/vstudio/device/prodinfo.asp,
2002-08-05

Macromedia Shockwave Player – White Paper,
http://www.macromedia.com/software/shockwaveplayer/whitepaper,
2002-08-05

Other

Products,
http://www.aidii.com/home2/product/casino.htm,
2002-08-2002

Games,
http://www.flux2game.com/games.htm?code=jack,
2002-08-2002