https://www.google.com/a/cpanel/guttman.ort.org.il/images/logo.gif?service=google_default

**מערכת גישה אוטומטית לאתרים**

**מבוססת SSO**

*תיק פרויקט*

***אליזבת לנגרמן***

ת.ז: 208568675

בית ספר: אורט גוטמן נתניה

מורים מנחים: מיכאל צ'רנובילסקי

גד רוזנטל

תאריך הגשה: 05.2016

**תוכן עניינים**

מבוא ורקע לפרויקט.................................................................................... 3

המטרות המרכזיות..................................................................................... 4

דרישות המערכת..................................................................................... 4

תיאור המוצר............................................................................................. 5

סביבת העבודה......................................................................................... 8

ארכיטקטורת המערכת................................................................................ 9

פירוט מאגרי המידע (ומסד הנתונים)............................................................. 12

מדריך התקנת המערכת.............................................................................. 14

ממשק המשתמש....................................................................................... 15

תהליך מחקר ופיתוח.................................................................................. 18

בבליוגרפיה............................................................................................... 21

קוד הפרויקט............................................................................................. 22

**מבוא ורקע לפרויקט**

כולנו רשומים כיום למספר רב של אתרים שנותנים לנו שירותים שונים. לדוגמא: כתובת אימייל אחת או יותר, גישה לחשבון הבנק שלנו דרך האינטרנט, קניות אונליין בחנויות שונות, חשבון פייסבוק, ושירותים רבים נוספים שמאפשרים להירשם אליהם כמנוי באמצעות שם משתמש וסיסמא.

קל מאוד לשכוח שמות משתמש וסיסמאות כאשר אנחנו בוחרים עבור כל שירות צירופים אחרים. עם זאת, כדי לשמור על הפרטיות שלנו ולהקטין את הסיכונים, אנחנו מעדיפים לא להשתמש באותה סיסמא, אלא להחליף אותה בכל אתר. לרוע המזל, כך קל מאוד לשכוח את אחת הסיסמאות או להתבלבל ביניהן. בנוסף, עלולה להיות אפשרות של Phishing. כלומר, אתרים מדומים שמתחקים אחר אתרי אינטרנט מוכרים, ומפנים את המשתמש לאתרים מזוייפים. כך הם עלולים לגנוב סיסמאות ומידע פרטי של המשתמשים.

הפרויקט שאני מפתחת מטרתו לתת מענה לבעיה זו, ע"י בניית מערכת הפועלת בשיטת SSO (Single Sign On). זוהי שיטת הזדהות של משתמש המאפשרת הזדהות אחת כלפי מספר מערכות מידע, ללא צורך בהזדהות נפרדת עבור כל מערכת.

המוצר שאני מפתחת אמור לתת למשתמש מסויים אפשרות גישה לאתרים שונים ללא צורך בסיסמאות רבות. המשתמש יוכל להתחבר למערכת (באמצעות סיסמא אחת בלבד) והיא תאפשר לו להתחבר לשירותים שתומכים במוצר זה ללא צורך בשם משתמש וסיסמא ייחודיים. כלומר, במקום להתחבר כרגיל, המשתמש יבחר באופציה המתאימה ויקבל גישה לחשבון שלו באופן אוטומטי.

**המטרות המרכזיות**

המטרות המרכזיות של הפרויקט שאני מפתחת הן:

* לאפשר גישה ישירה, מהירה ואוטומטית לאתרי אינטרנט שתומכים באפשרות זו.
  + הגישה תהיה באמצעות התחברות אחת בלבד, שתאפשר למשתמש להתחבר לחשבון שלו במספר אתרים ללא צורך באמצעי זיהוי רבים.
* להפוך את השימוש בסיסמאות לקל ונוח יותר.
  + להקל על המשתמש שלא יצטרך לזכור סיסמאות רבות ושונות.
  + ישנן פניות רבות של משתמשים שנושא סיסמאות ושחזורן. לכן מערכת כזו יכולה לצמצם עלויות של צוותי מחשוב שמטפלים בפניות אלה.
* להגדיל את האבטחה ע"י הקטנת הסיכוי לגניבת סיסמאות באמצעות PHISHING (אתרים מדומים).

**דרישות המערכת**

דרישות פונקציונאליות:

* לאפשר התחברות לתוכנית הראשית, באמצעות שם משתמש וסיסמא.
* לאפשר למשתמש שהתחבר גישה לתוכניות משניות התומכות באפשרות זו, ללא צורך בשם משתמש וסיסמא.
* להציג למשתמש המחובר את האתרים שבאפשרותו להתחבר אליהם באמצעות מערכת זו.
* לתת למשתמש אפשרות להתנתק מהמערכת.
* אפשרות חיבור של מספר משתמשים בו זמנית (multi-client).

דרישות ממשקים:

* כל אחת מהתוכניות (ראשית ומשנית) תכיל ממשק גרפי עבור המשתמש (GUI).
* תקשורת בין התוכנית הראשית לתוכניות המשניות שיאפשר התחברות.
* תקשורת בין השרת הראשי ללקוחות המתחברים אליו באמצעות אינטרנט.

דרישות לא פונקציונאליות:

* אבטחה: שמירה מאובטחת על סיסמאות והעברה מוצפנת של מידע.
* אפשרות צפייה בסטטוס המשתמשים לצרכי תחזוקה.

דרישות מסד נתונים:

* שמות המשתמש והסיסמאות לתוכנית הראשית יישמרו בDatabase של 3SQLITE.

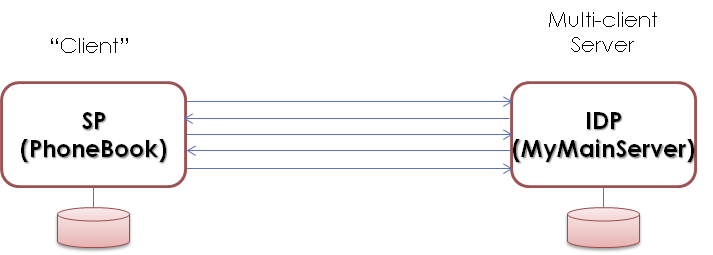
**תיאור המוצר**

המערכת שאני בונה מורכבת משרת מרכזי, "השרת הראשי" שכיניתי MyMainServer, וכן משרת נוסף לדוגמא, "שרת משני" (SP). השרת הראשי מספק את שירות ה- SSO שאני מעוניינת לבנות במערכת שלי, ואליו השרת המשני מדמה אתר אינטרנט כלשהו שאליו מתחברים משתמשים רשומים, והוא משתמש בשירות שאני מציעה.

תהליך התחברות משתמש למערכת מתבצע כך:

ראשית, שרת שמעוניין ליצור קשר עם המערכת, מתחבר לשרת הראשי באמצעות socket. במסגרת התקשורת ביניהם (client-server), מקבל השרת המשני (ה- SP) מזהה ייחודי ומפתח ייחודי שדרכו תתבצע הצפנת התקשורת בינו לבין השרת הראשי. הנתונים מוצפנים בעזרת הצפנה א-סימטרית, RSA. כמו כן, הוא שולח אל השרת הראשי את כתובתו, על מנת שבהמשך יוכל להפנות משתמשים בחזרה אליו (באמצעות redirect).

לאחר חילופי המידע, השרת החדש נרשם במסד הנתונים של השרת הראשי.



connect

spName@redirectPath

OK Send name, redirect path

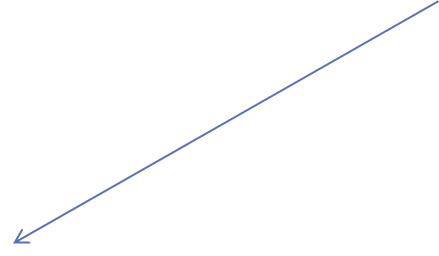
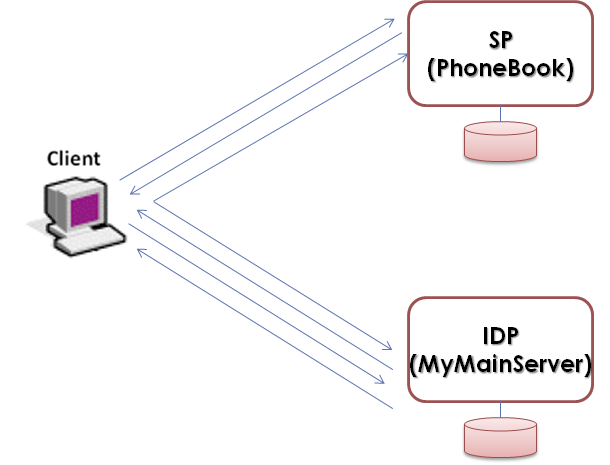
ID@key

OK End Connection

כעת, משתמש שמבקש להתחבר לשרת ה-SP (לדוגמא, השרת PhoneBook), יכול לבחור באופציה המתאימה ולהתחבר באמצעות MyMainServer, השרת הראשי.

בבחירת האופציה, המשתמש יועבר (ע"י redirect) לשרת הראשי, ויבצע מולו את ההתחברות עם שם משתמש וסיסמא. חשוב לציין ששם המשתמש והסיסמא משמשים רק לשם התחברות ל-MyMainServer ולא לשרת של אתר האינטרנט שהמשתמש ביקש להתחבר אליו מלכתחילה.

השרת הראשי ימצא את המשתמש במסד הנתונים, ויעביר את ה- ID שלו כפי שהוא רשום בשרת ה-SP. הוא יפנה את המשתמש בחזרה אל השרת המשני, אל העמוד השייך לו, העמוד שמכיל את הפרטים של משתמש זה.



(6) User Profile

(5)

(5) Redirect back to SP server

(4) Credentials

(3) Login Page

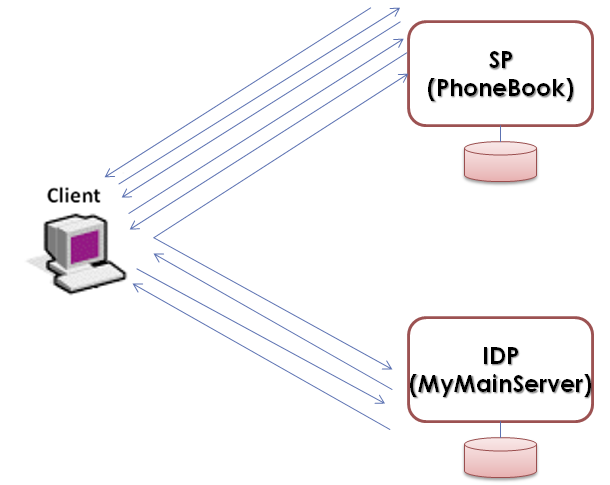
(2)

(1) User wants to connect to SP server

(2) Redirect to MyMainServer

באם משתמש חדש רוצה להירשם לאתר המשתמש במערכת, הוא נכנס כרגיל להרשמה של אותו אתר. המשתמש מופנה אל דף ההרשמה של השרת הראשי, ונרשם אליו באמצעות שם משתמש וסיסמא. לאחר שנוסף את מסד הנתונים של השרת הראשי, מופנה המשתמש שוב אל השרת המשני, אל דף ההרשמה שלו. בדף זה יוכל להכניס נתונים שונים כדי להשלים את הרשמתו לאתר שביקש להירשם אליו.

במילים אחרות, יבצע את כל ההרשמה כמו שהיא מול שרת ה- SP, אך כל הטיפול בהתחברות ובסיסמאות יישאר בידי השרת הראשי, MyMainServer.



(6) SP user registration page

(7) User info

(8) User Profile page

(1) User wants to register to SP server

(2) Redirect to MyMainServer

(5)

(2)

(3) Registration Page

(4) Credentials

(5) Redirect back to SP server

**סביבת העבודה**

מנוע הפרויקט נכתב בשפת פייתון (Python 2.7), בסביבת העבודה PyCharm.

הקוד עבור עמודי ה-WEB נכתב בשפת HTML, בסביבת העבודה Notepad++.

הפרויקט נוסה ונבדק תחת מערכת ההפעלה Windows ובפרט נבדק במערכות: Windows 7, Windows 8.

בפרויקט יש שימוש במודולים הבאים:

SQLite3 - מודול שמאפשר יצירת Database וניהולו

Flask - מודול שמאפשר ניהול שרת WEB הכתוב בפייתון

PyCrypto - מודול להצפנה בשפת פייתון

ConfigParser - מאפשר קריאה של קובץ קונפיגורציה .cfg

Socket - מאפשר תקשורת בין שרת ללקוחות

Thread - מודול שבאמצעותו ניתן ליצור שרת Multithreading, ובאופן כללי לגרום

למספר קטעי קוד לרוץ בו זמנית

String - מודול שמאפשר עבודה רחבה עם מחרוזות

Sys - מאפשר ביצוע פעולות בסיסיות של המערכת (system)

Random - באמצעותו ניתן ליצור תווים ורצפים רנדומליים

Base64- מאפשר קידוד הודעות

Hashlib - מאפשר ליצור hash של מחרוזת (למשל לצורך שמירת סיסמאות)

**ארכיטקטורת המערכת**

הפרויקט מורכב משני שרתים מרכזיים: השרת הראשי ()IDP, ושרת לדוגמא שיתמוך באפשרות שאני מפתחת (SP). בשני השרתים תהיה תמיכה לחיבור לקוחות במקביל- multithreading.

*השרת הראשי(IDP) - חלוקה למודולים:*

1. **New User Registration**

מודול שאחראי על הרשמה של משתמשים חדשים, בין אם לשרת SP מסויים או לשרת המרכזי- IDP בעצמו.

* מקבל מזהה של הSP שממנו נשלחה הבקשה
* מקבל ID של המשתמש כפי שנרשם בשרת ה SP ושומר אותו
* מבקש פרטי הרשמה או התחברות מהמשתמש
* בודק האם המשתמש קיים, או שהוא חדש לחלוטין
* בהתאם, שומר את המידע החדש שנוסף על משתמש זה
* Redirect בחזרה לשרת הראשון (SP) + הודעת הצלחה

1. **User Login**

מודול שתפקידו לאפשר למשתמש להתחבר לשרת הראשי ומספק את פרטי ההתחברות של המשתמש לSP.

* מקבל מזהה של הSP שממנו נשלחה הבקשה לזיהוי לקוח
* מבקש מהמשתמש להכניס פרטי זיהוי
* מוודא את קיום המשתמש (בהתאם לDatabase)
* ממפה בין המשתמש לבין ה ID שלו בשרת ה SP המבקש
* Redirect בחזרה לשרת הראשון (SP) + שליחת user SP ID ל SP.

1. **SP Registration**

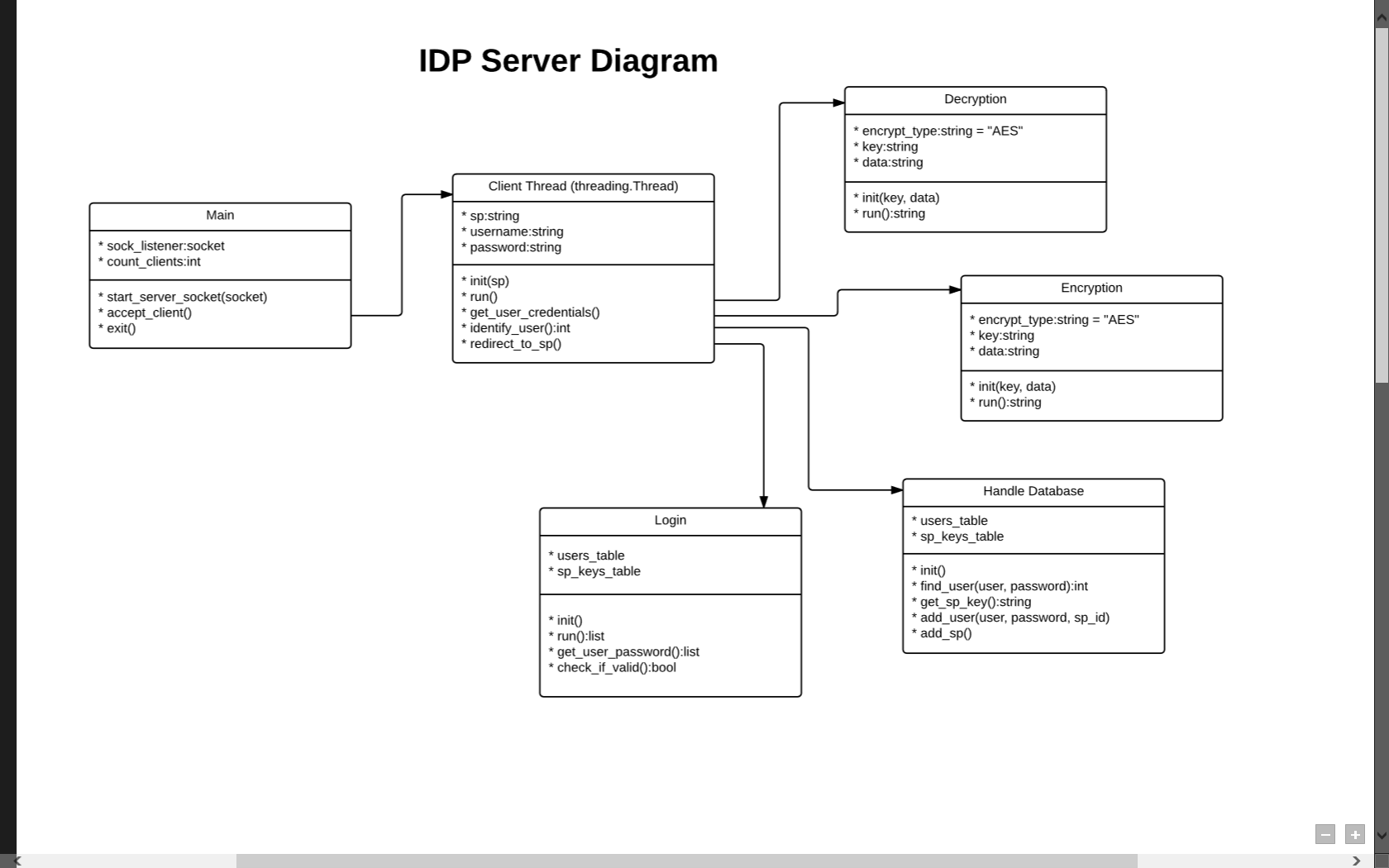
מודול שתפקידו תקשורת ראשונית עם שרת שמתחיל להשתמש במערכת.

* מקבל בקשה משרת הSP החדש
* מבקש ממנו פרטים חיוניים לתקשורת ביניהם (לדוגמא מידע כללי, הכתובת שאליה לעשות redirect, רשימת מזהים של המשתמשים הרשומים לשרת זה)
* מנפיק מפתח להצפנה סימטרית בין שני הצדדים ושולח לSP.

1. **Handle Database**

המודול מכיל את מסד הנתונים ואת הכלים לעבודה מול מסד הנתונים.

* אפשרות הוספת משתמש
* חיפוש משתמש והחזרת ה ID שלו בהתאם לSP המבקש
* חיפוש SP ואפשרות להוספה



*השרת המשתמש במערכת(SP) - חלוקה למודולים:*

1. **Login**

מודול המאפשר למשתמש כלשהו להתחבר לשרת.

* אפשרות התחברות באמצעות שם משתמש וסיסמא או באמצעות שרת ה IDP.
* Redirect לשרת המרכזי IDP + שליחת מזהה ייחודי מוצפן של הSP.
* הצגת נתוני המשתמש אחרי ההתחברות, כרגיל.

1. **Register**

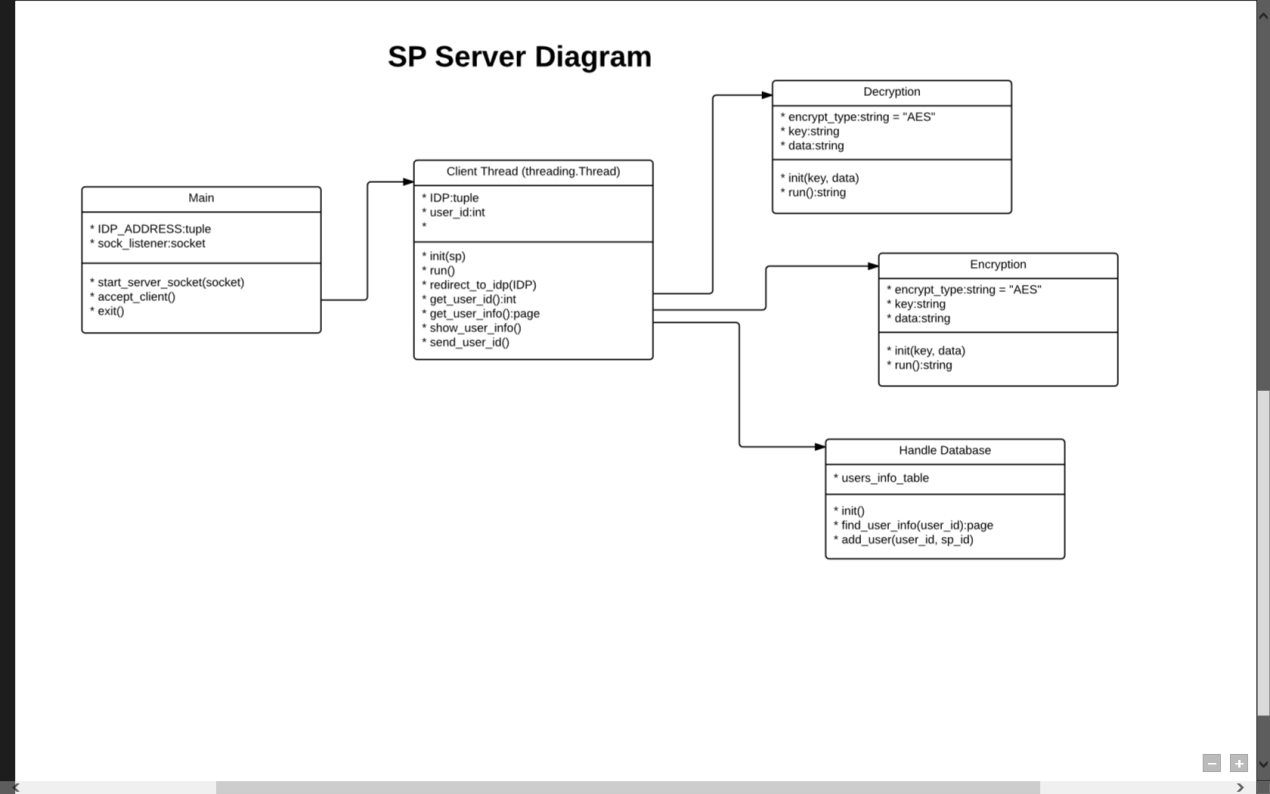
מודול שאחראי על הרשמת משתמש חדש לאתר, והתקשורת עם השרת הראשי במצב זה.

* Decrypt
* Decode
* במקרה של הצלחה, Redirect למודול User Page

1. **Handle Database**

המודול מכיל את מסד הנתונים ואת הכלים לעבודה מול מסד הנתונים.

* אפשרות הוספת משתמש
* חיפוש משתמש והחזרת העמוד הפרטי שלו בהתאם ל userID



**פירוט מאגרי המידע (מסד הנתונים)**

המערכת יוצרת מסד נתונים בצד השרת המרכזי MyMainServer, וגם בצד שרת המשני (שרת ה- SPלדוגמא : Phonebook).

במסד הנתונים בצד השרת הראשי MyMainServer ישנן שתי טבלאות:

1. טבלה בשם **Users** שמאחסנת את הנתונים של כלל המשתמשים הרשומים למערכת. מבנה הטבלה הוא:

|  |  |  |
| --- | --- | --- |
| **שם העמודה** | **סוג** | **ערך** |
| ID | INT | מזהה של השתמש |
| username | TEXT | שם משתמש |
| passw | TEXT | hash של סיסמת המשתמש |
| spID | INT | מזהה השרת המשני שאליו שייך המשתמש |
| spUserID | INT | מזהה המשתמש כפי שרשום בשרת המשני |

1. טבלה בשם **SPs** שמאחסנת את הנתונים של כלל השרתים המשניים (SP) הרשומים למערכת. מבנה הטבלה הוא:

|  |  |  |
| --- | --- | --- |
| **שם העמודה** | **סוג** | **ערך** |
| SPID | INT | מזהה של השרת |
| details | TEXT | פרטים של השרת |
| redirectPath | TEXT | כתובת של השרת המשני כדי לבצע redirect |
| key | TEXT | מפתח מוצפן לשימוש בתקשורת מול שרת זה |

במסד הנתונים בצד השרת המשני ישנה טבלה אחת בשם **UserProfiles**:

|  |  |  |
| --- | --- | --- |
| **שם העמודה** | **סוג** | **ערך** |
| ID | INT | מזהה של השתמש |
| name | TEXT | שמו של המשתמש |
| age | INT | גילו של המשתמש |
| phoneNum | TEXT | מספר הטלפון של המשתמש |

**מדריך התקנת המערכת:**

המערכת עושה שימוש במודולים SQLite3, Flask, PyCrypto, שדורשים התקנה מיוחדת.

* התקנת ספריית PyCrypto לפייתון Python 2.7 (32-bit): <http://www.voidspace.org.uk/downloads/pycrypto26/pycrypto-2.6.win32-py2.7.exe>
* התקנת מודול Flask לפייתון Python 2.7 (32-bit), באמצעות pip:

להתקין pip ולהריץ את הפקודה: pip install flask

התקנת pip:

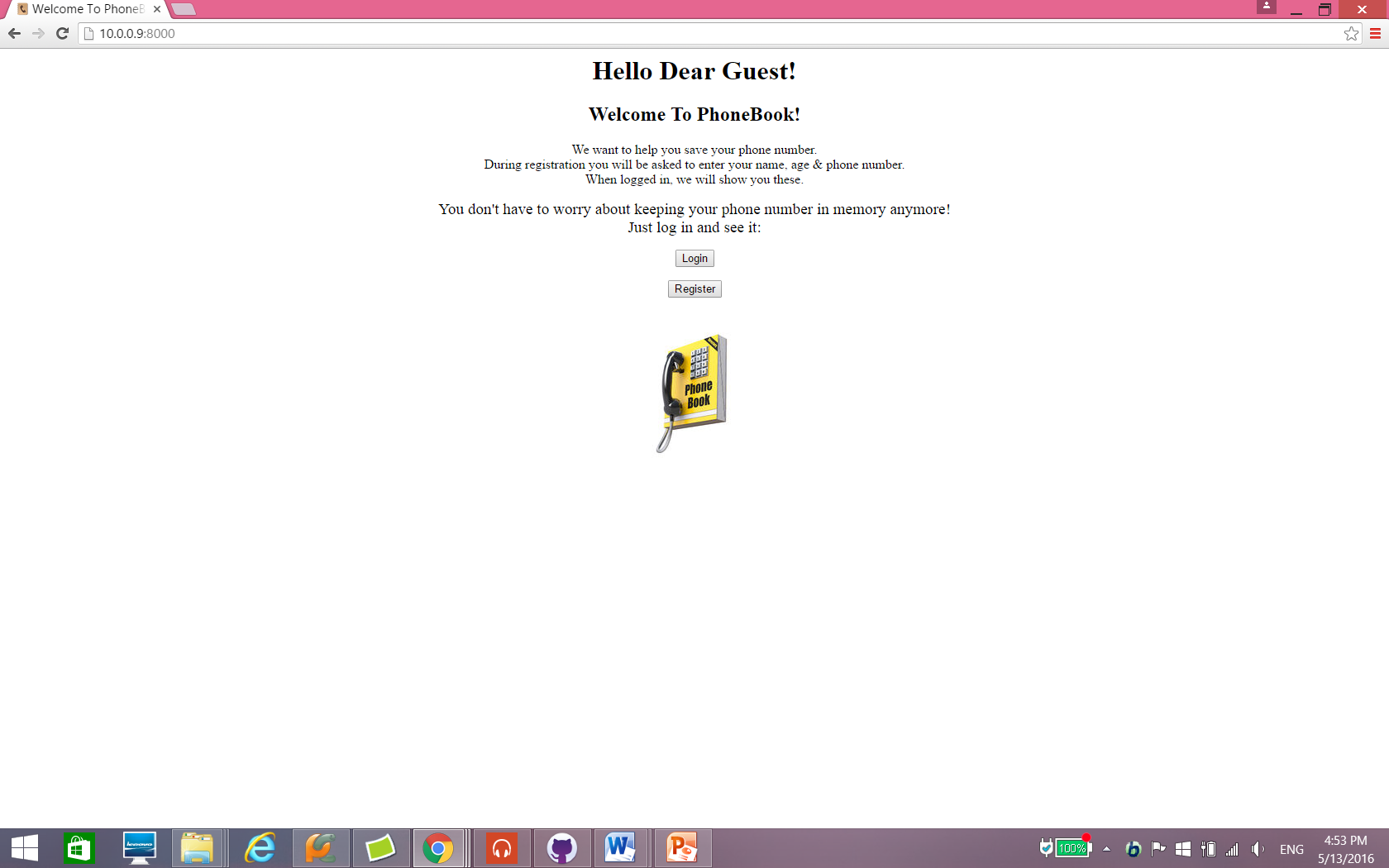
[https://pip.pypa.io/en/latest/installing](https://pip.pypa.io/en/latest/installing/)

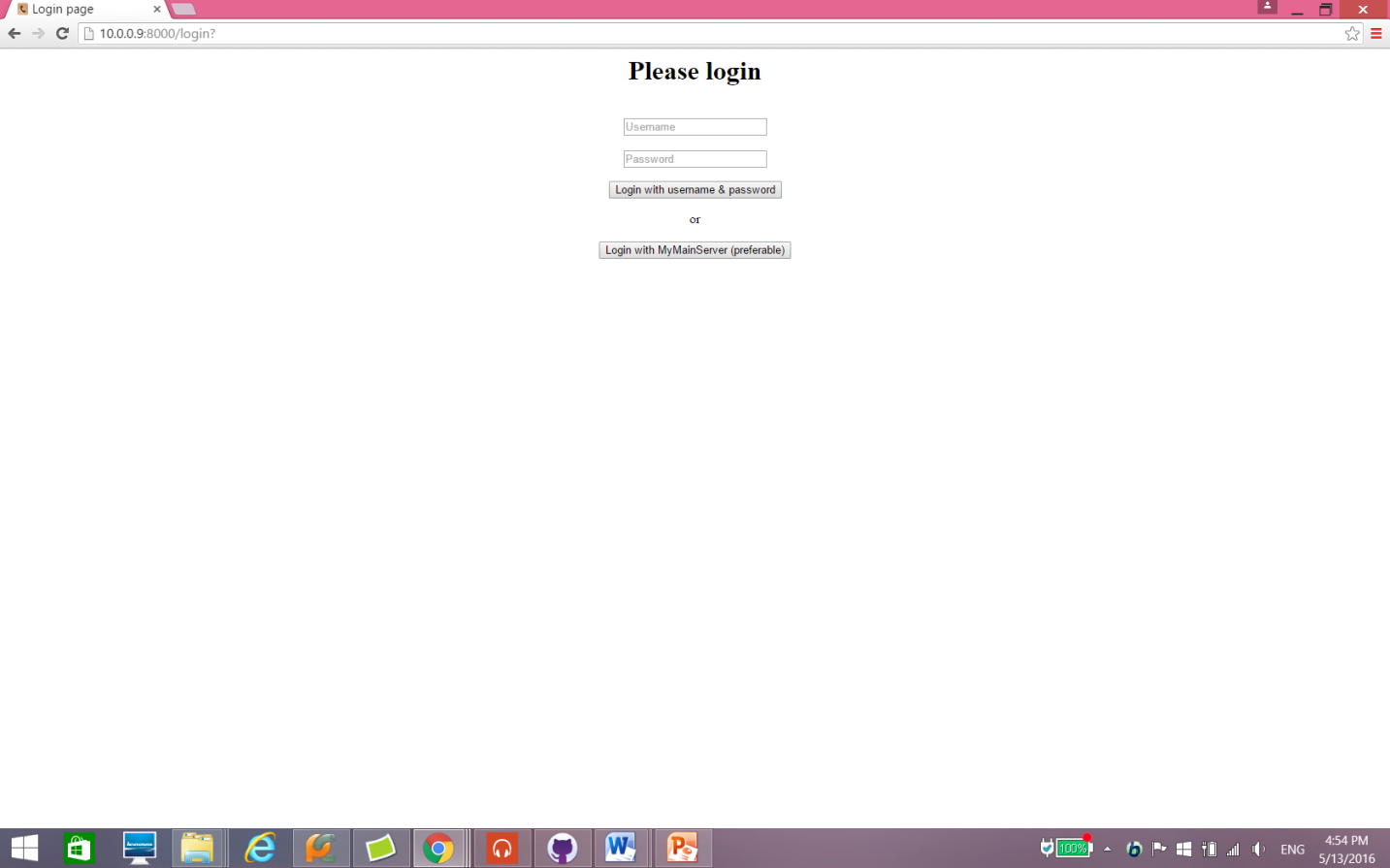
* התקנת מודול sqlite3 לפייתון Python 2.7 (32-bit): <http://www.tutorialspoint.com/sqlite/sqlite_installation.htm>

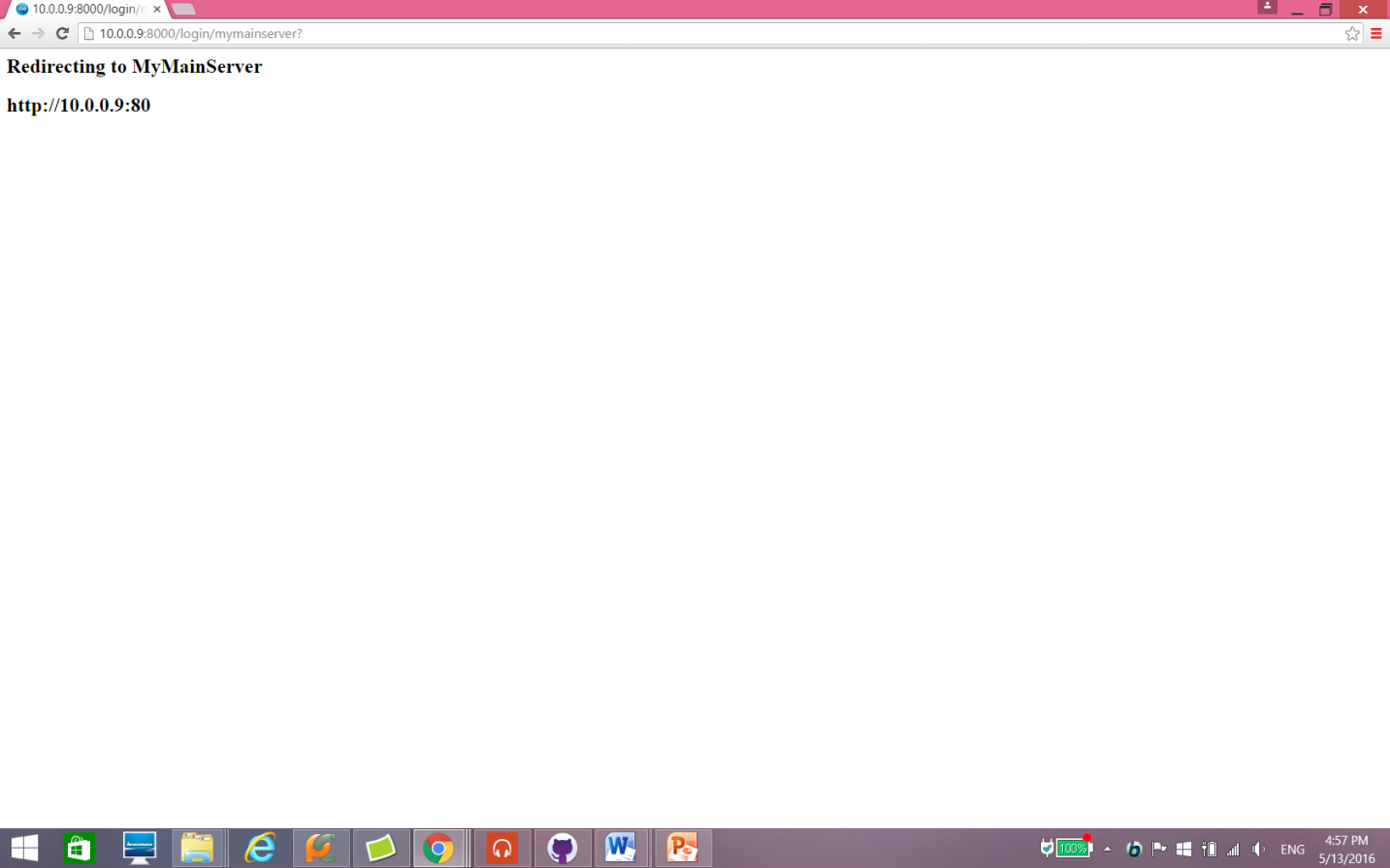
הפרויקט כולו שמור באמצעות GitHub, וניתן למצוא אותו בכתובת <https://github.com/ELLE209/MyProject/tree/master/MyProject>

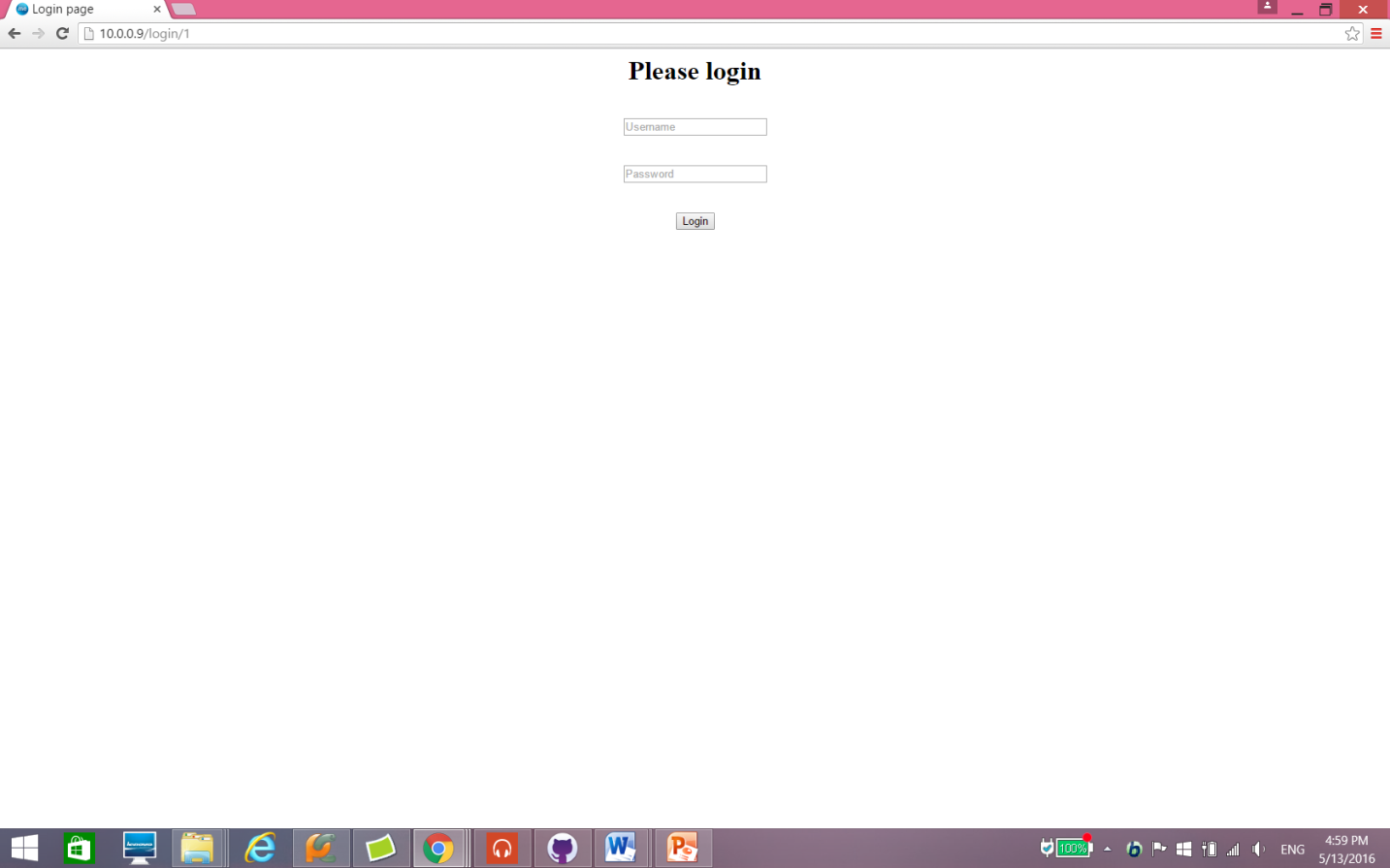
יש לשנות את הנתונים הקבועים, לפי מיקום המערכת, בקבצי הקונפיגורציה של שני השרתים: main\_config.cfg, sp\_config.cfg.

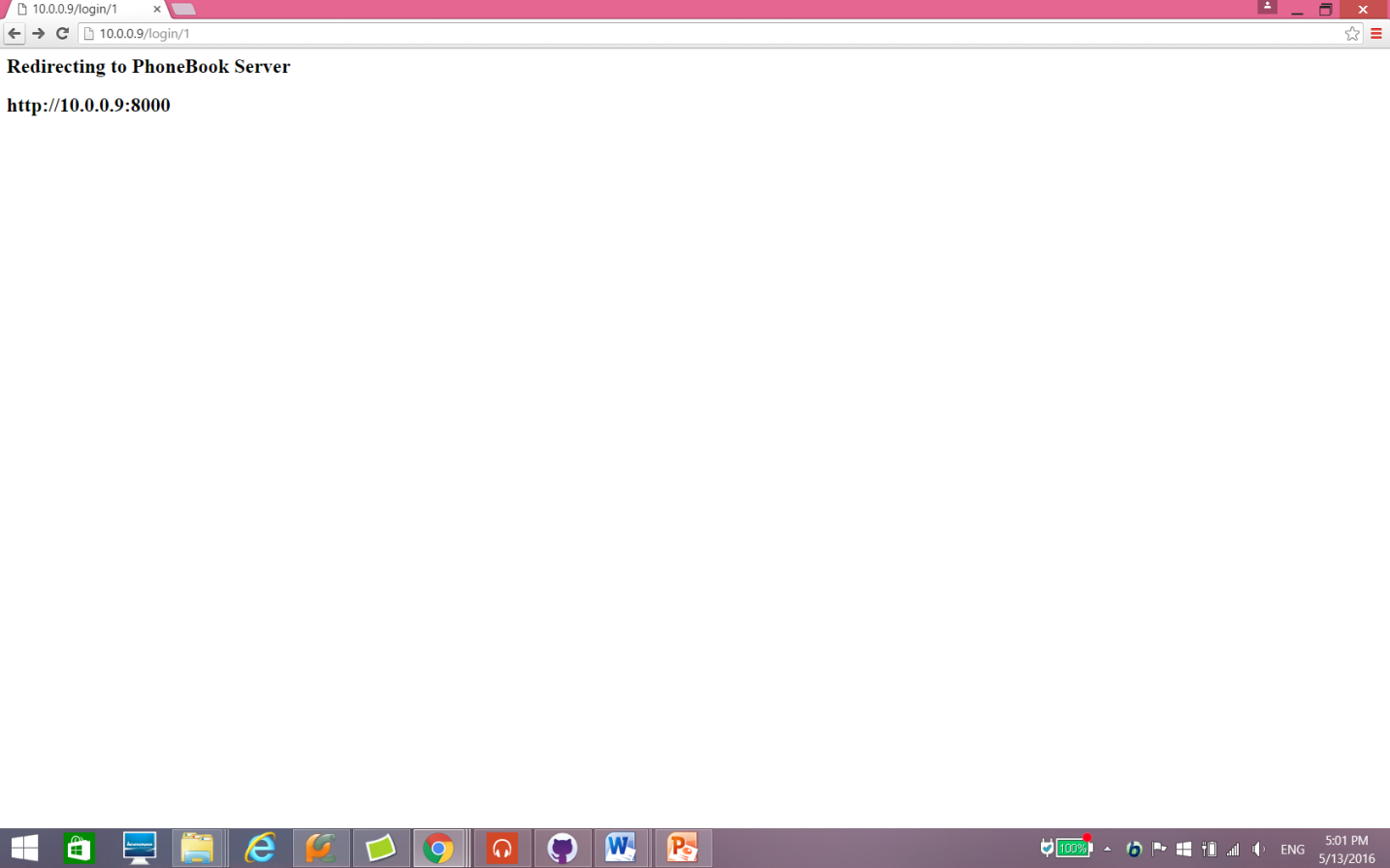
**ממשק המשתמש**

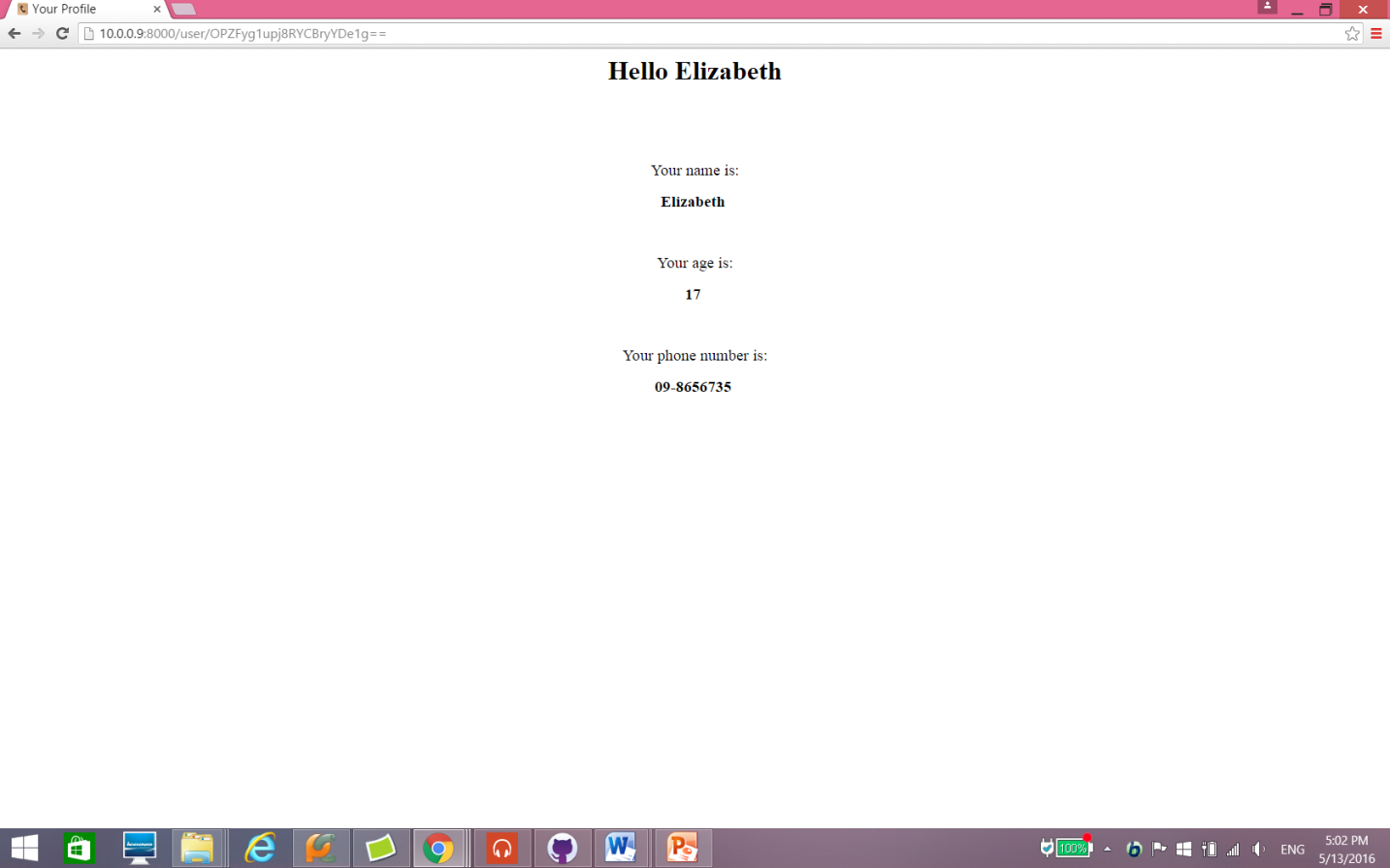


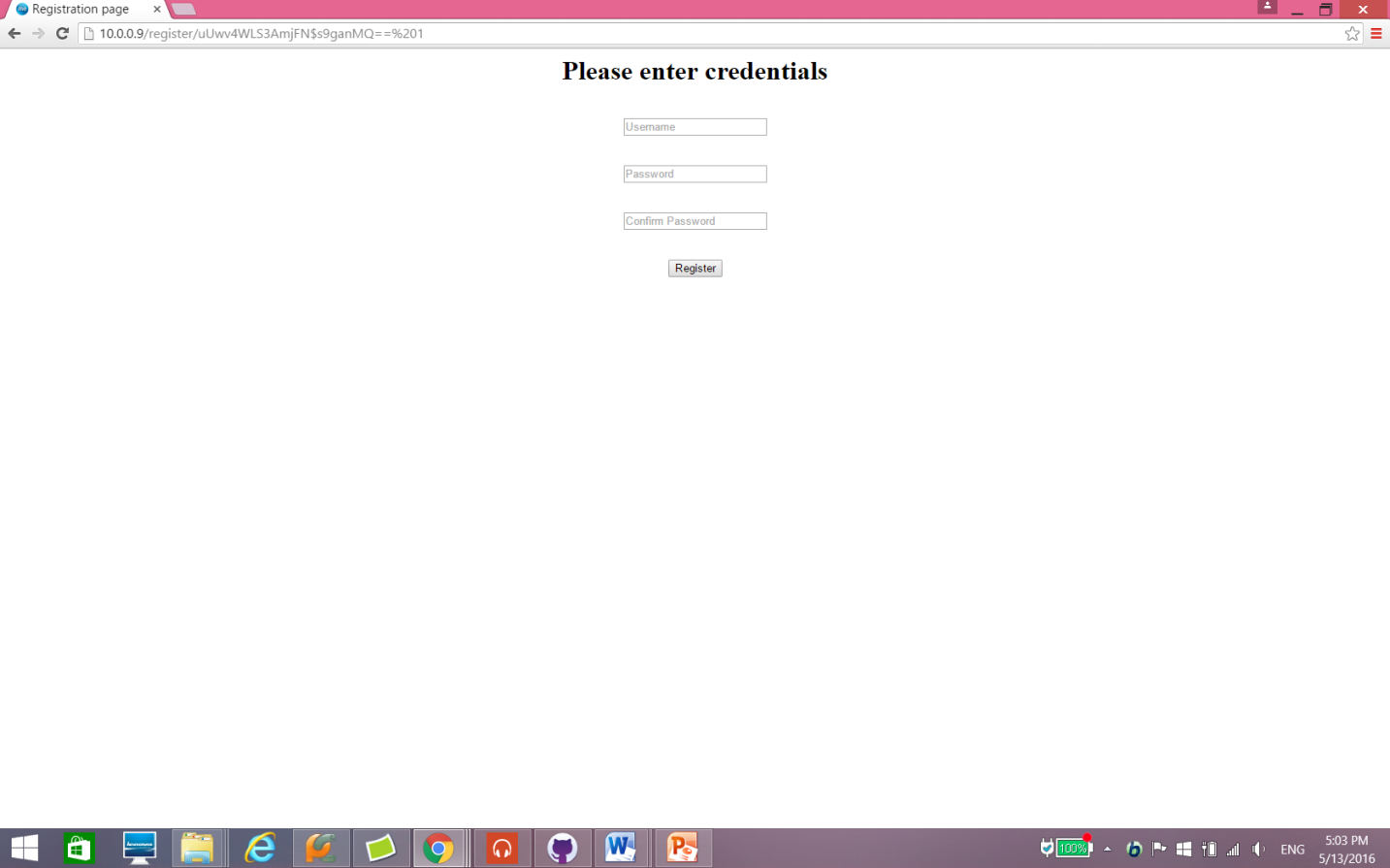


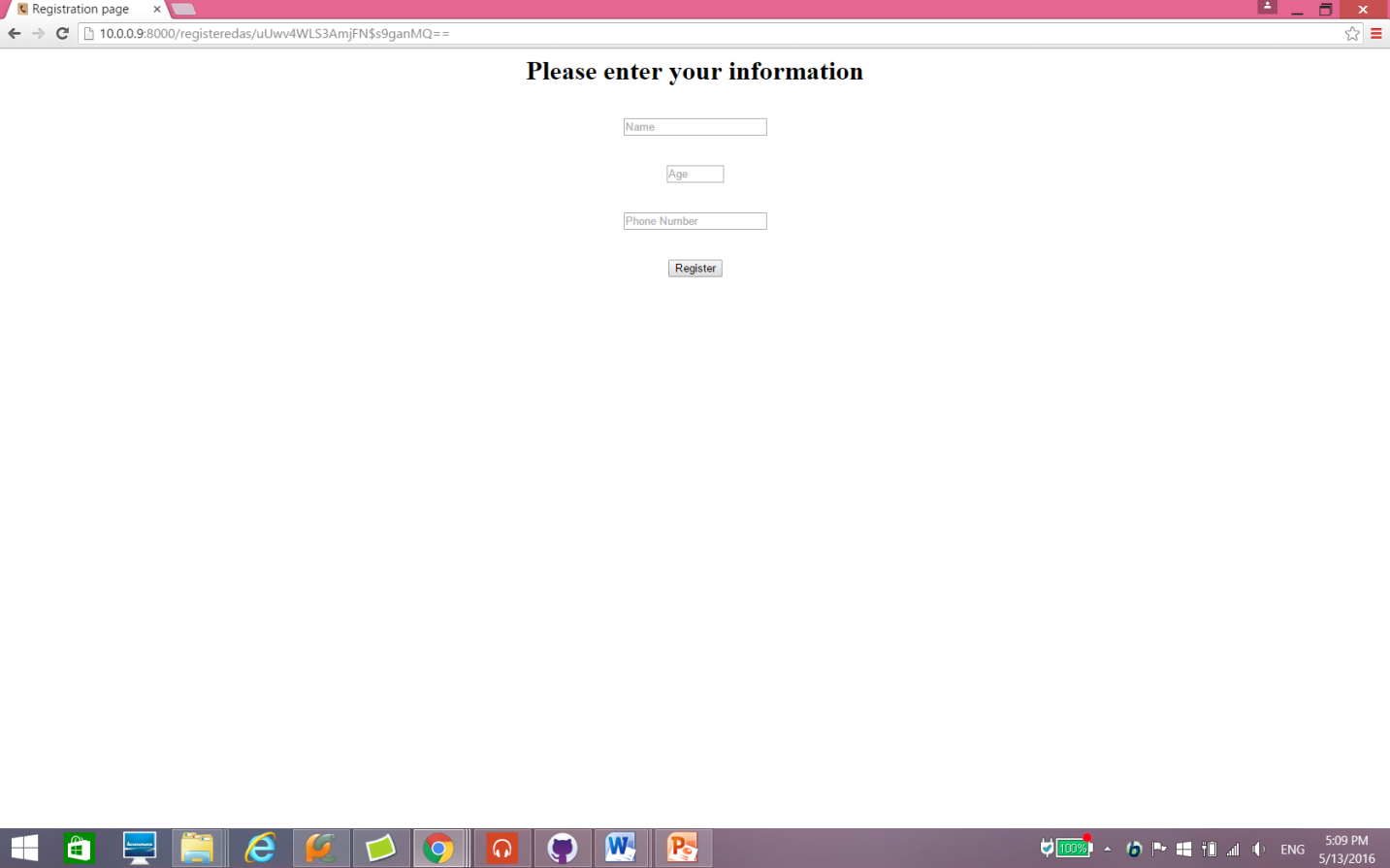












**תהליך מחקר ופיתוח**

המערכת שאני בונה בפרויקט זה מטרתה יצירת אפשרות SSO. יש מספר שיטות ליצור זאת, ולאחר מחקר על SAML ועל 2OAUTH, החלטתי שהמבנה שלSAML מתאים יותר למבנה המערכת שאני רוצה לפתח.

התחלתי לחפש מודול שיסייע לי ביצירת מערכת שפועלת לפי דוגמת SAML. מצאתי מספר מודולים, ומתוכם בחרתי את pysaml2. רציתי להשתמש תחילה במודול זה משום שהוא מאפשר לעבוד עם SAML, ליצור IDP, SP וכו'.

התחלתי ללמוד את השימוש במודול, אך לאחר מחשבה מעמיקה הבנתי שאעדיף לא להשתמש בו, אלא אוכל לבנות מערכת שצורת פעולתה דומה למבנה המוגדר בSAML, ללא שימוש במודול מיוחד לשם כך.

תיארתי מבנה מערכת חדש, שבו לא יהיה צורך במודול זה. שרטטתי גם תרשים זרימה שמתאר באופן רציף את תהליך פעולת המערכת. פירטתי בתרשים מה תבצע כל אחת מהתוכנות שהן חלק מהפרויקט שלי.

באמצעות התרשים התחלתי לכתוב את הקוד בצורה מסודרת. ראשית התחלתי מכתיבת הService Provider כולל ממשק המשתמש שלו. לאחר שזה מוכן והגרסא הראשונית שלו פועלת כראוי, התקדמתי לכתיבת הגרעין של המערכת – ה Identity Provider.

לאחר שבניתי מערכת הכוללת ממשק #C, הבנתי שעדיף יהיה לבנות את הפרויקט כך שהשרתים יהיו שרתי web. כלומר, הלקוח יהיה הדפדפן- באמצעות הדפדפן יפנה המשתמש אל שרת(SP) כלשהו, שיחזיר לו redirect לשרת ה IDP. שיטה זו נפוצה ויעילה יותר לדעתי מאשר אפליקציות בודדות עם ממשק #C, גם מפני שאפשר להפעיל שרת ולהתחבר אליו ממחשבים שונים באמצעות הדפדפן.

באמצעות תרשימים לתהליכים שארצה לממש בפרויקט שלי, הגדרתי סופית את המבנה והפרוטוקולים כאשר מדובר בשרתי web. במהלך בניית התרשימים נתקלתי במספר שאלות:

* *באחד מתהליכים אלו, אמור שרת הIDP להפנות את הלקוח בחזרה לשרת הSP. כיצד יפנה אותו בחזרה לשרת הנכון?*

ה SP צריך לשלוח לIDP מידע בנוגע לזהותו. דרך אחת לעשות זאת היא באמצעות שדה הhttp referrer. דרך נוספת היא באמצעות שליחת מזהה ייחודי של השרת המבקש, עם שליחת redirect. הדרך השלישית היא שליחת המזהה בנפרד, ישירות לשרת הIDP. אמנם בדרך זו המידע לא עובר דרך המשתמש ולכן התקשורת מאובטחת יותר, אך יש צורך בתקשורת נוספת עם השרת המרכזי, שעלולה להיות לא מתואמת מבחינת זמנים ונוחות. בחרתי להשתמש בדרך של שליחת מזהה מוצפן עם שליחת הredirect.

* *באיזה סוג הצפנה כדאי להשתמש?*

ראשית חשבתי להשתמש בהצפנה א-סימטרית, משום רמת האבטחה הגבוהה יותר. בהצפנה סימטרית יש צורך בהעברת מפתח, ולכן יורדת מידת האבטחה. אבל, במבט מעמיק יותר, שני הצדדים צריכים להצפין ולפענח מידע במערכת זו. לכן, יהיה יעיל יותר להשתמש בהצפנה סימטרית בכל זאת.

* *מתי ליצור וכיצד להעביר את מפתח ההצפנה בין שני השרתים?*

עבור כל שרת SP שמתחבר אליו, נחוץ ל IDP מפתח קבוע. לכן, צריך להיות מנגנון של רישום SP, שבמהלכו יחליפו פרטים שני השרתים, וגם את מפתח ההצפנה (באמצעות הצפנה א-סימטרית). כך אין צורך בהחלפת מפתחות הצפנה בזמן שלקוח מתחבר, וגם לא בכל פעם מחדש.

* *כיצד לשמור את המידע על המשתמשים הרשומים?*

לצורך שמירת שמות המשתמשים והסיסמאות אשתמש ב SQLite3. כך גם לצורך התאמת מזהה המשתמש בין השרתים ולשמירת מידע של כל משתמש.

* *איך אבנה שרת web?*

התחלתי ללמוד את השימוש במודול flask לפייתון. מודול זה מאפשר בניית שרת web בשפת פייתון. לצורך השימוש בו הבנתי שאצטרך לכתוב מספר קבצי HTML קצרים, שיהוו את עמודי ה WEB של השרתים.

לאחר שהתחלתי לעבוד עם המודולים Flask, SQLite3, במהלך כתיבת העבודה, נתקלתי בשאלות נוספות:

* *איך לקשר את קבצי ה-HTML אל מנוע הפייתון?*

לצורך כך מצאתי את הפעולה render\_template, ואת אופן השימוש בה. פעולה זו מאפשרת הפעלה של template עם פרמטרים במידת הצורך, מתוך קובץ הפייתון.

* *כיצד אוכל לשמור עבור כל SP את אובייקט ההצפנה שלו?*

במהלך הפיתוח גיליתי כי לא ניתן לשמור ב Database אובייקט הצפנה, כמו מופע של המחלקת Encryption שכתבתי, שאחד מהמאפיינים שלו הוא אובייקט AES. החלטתי שאשמור את המפתח בלבד במסד הנתונים, ואצור באמצעותו אובייקט הצפנה בעת הצורך.

* *כיצד אשמור את המפתחות בצורה מאובטחת יותר?*

לכל שרת SP שרשום למערכת צריך להיות מפתח ייחודי וקבוע לשם התקשורת עם השרת הראשי. הפתרון שבחרתי הוא שמירת המפתחות מוצפנים במסד הנתונים, כאשר מפתח ההצפנה שמור בתוך קוד התוכנית.

* *מה יקרה אם מתקבל '/' אחרי ההצפנה?*

בתהליך ההצפנה, אחד התווים שיכול להתקבל הוא '/'. כתוצאה מהעברת המחרוזת המוצפנת, נוצרת שגיאה שכן התו '/' הוא תו מפריד קבוע בכתובות WEB. כדי לפתור את הבעיה, חיפשתי כיצד להגדיר את התווים שיתקבלו לאחר ההצפנה. לא מצאתי שיטה טובה לעשות זאת, ולכן עברתי לרעיון אחר. החלטתי להחליף כל תו '/' שיתקבל במחרוזת המוצפנת בתו '$'. תו זה אינו חלק מן התווים המתקבלים לאחר ההצפנה. ולכן החלטתי להחליף את כל המופעים של '/' ב- '$' לפני העברת המחרוזת המוצפנת ב URL ולהחליף את התווים בחזרה לפני פענוח המחרוזת.

* *כיצד להציג למשתמש כי מתבצע redirect?*

ממשק המשתמש הוא פשוט מאוד, והמשתמש אינו רואה את כל העבודה המתבצע "מאחורי הקלעים". כדי להמחיש למשתמש את המעברים משרת אחר למשנהו, יצרתי עמוד נוסף, שיופיע למספר שניות ויציג את ה-redirect שמתבצע.

**בבליוגרפיה**

* MSDN, הרשת הרשמית של Microsoft למפתחים: <http://msdn.microsoft.com/library>
* הפורום הגדול ביותר לשאלות ותשובות בנושאי תכנות: [http://stackoverflow.com](http://stackoverflow.com/)
* התיעוד הרשמי של Python: [https://docs.python.org/2.7](https://docs.python.org/2.7/)
* אתר ללימוד SQLite: <http://www.tutorialspoint.com/sqlite>
* להורדת Python 2.7 (32-bit):

<https://www.python.org/ftp/python/2.7/python-2.7.msi>

* התקנת ספריית PyCrypto לפייתון Python 2.7 (32-bit): <http://www.voidspace.org.uk/downloads/pycrypto26/pycrypto-2.6.win32-py2.7.exe>
* התקנת מודול Flask לפייתון Python 2.7 (32-bit), באמצעות pip:

להתקין pip ולהריץ את הפקודה: pip install flask

* התקנת pip:

[https://pip.pypa.io/en/latest/installing](https://pip.pypa.io/en/latest/installing/)

* התקנת מודול sqlite3 לפייתון Python 2.7 (32-bit): <http://www.tutorialspoint.com/sqlite/sqlite_installation.htm>

**קוד הפרויקט**

מצוף כל הקוד של הפרויקט שלי:

**MyMainServerWeb**

*# region ----------------------- ABOUT --------------------------  
"""  
################################################################  
# Created By: Elizabeth Langerman #  
# Date: 04/2016 #  
# Name: Identity Provider (IDP) Server =Main Server #  
# Version: 1.0 #  
# Windows Tested Versions: Win 8 32-bit, Win 7 32-bit #  
# Python Tested Versions: 2.7 32-bit #  
# Python Environment: PyCharm #  
################################################################  
"""  
# endregion  
  
# region ---------------------- IMPORTS -------------------------***from** DataBaseManager **import** \*  
**from** Encryption **import** \*  
**from** ThreadWithReturnValue **import** \*  
**from** flask **import** Flask, request, redirect, render\_template  
**import** ConfigParser  
**import** socket  
**import** thread  
**import** string  
**import** random  
**import** hashlib  
**import** sys  
*# endregion  
  
# region ------------------ CONFIGURATIONS ----------------------*config = ConfigParser.ConfigParser()  
config.readfp(open(**'main\_config.cfg'**)) *# path to config file***try**:  
 *# get constants from configurations file* PORT = int(config.get(**'General'**, **'port'**))  
 DB\_NAME = config.get(**'General'**, **'db\_name'**)  
**except** ConfigParser.NoSectionError **or** ConfigParser.NoOptionError, e:  
 **print** e  
 sys.exit(1)  
  
HOST = **'0.0.0.0'**MASTER = **b'Sixt33n Byt3 K3y'***# endregion  
  
# region ---------------------- GLOBALS -------------------------*app = Flask(\_\_name\_\_)  
server\_sock = socket.socket()  
*# endregion  
  
  
# region ------------------- URL\_FUNCTIONS ----------------------  
# route for MyMainServer login page*@app.route(**'/login/<sp>'**, methods=[**'GET'**, **'POST'**])  
**def** login(sp):  
 error = None  
 **if** request.method == **'POST'**:  
 **try**:  
 *# get username & password hash from html form* username = request.form[**'username'**]  
 hashed\_pass = hashlib.sha1(request.form[**'password'**]).hexdigest()  
 sp\_id = int(sp)  
  
 *# find user in database* query = **"SELECT spUserID FROM Users WHERE username='%s' AND passw='%s' AND spID=%d"** %\  
 (username, hashed\_pass, sp\_id)  
 data\_base\_manager = DataBaseManager(DB\_NAME)  
 sp\_user\_id = data\_base\_manager.exec\_query(query)[0]  
 data\_base\_manager.close\_connection()  
  
 *# redirect to user's profile page is SP* sp\_user\_id = get\_encrypt\_obj(sp\_id).encryptAES(str(sp\_user\_id))  
 addr=get\_sp\_url(sp\_id)  
 sp\_url = **'%s/user/%s'** % (addr, str(sp\_user\_id))  
 server = **"PhoneBook Server"  
 return** render\_template(**'RedirectPage.html'**, path=sp\_url, server=server, addr=addr)  
 *#return redirect(sp\_url)* **except** Exception, exc:  
 *# if user wasn't found* error = **'Invalid Credentials. Please try again.'  
 print** exc  
 **return** render\_template(**'LoginMain.html'**, error=error)  
  
  
*# route for registration page*@app.route(**'/register/<userid> <sp>'**, methods=[**'GET'**, **'POST'**])  
**def** register(userid, sp):  
 **print** userid  
 **print "sp: "** + sp  
 sp\_id = int(sp)  
  
 *# decrypt user ID with the key for this SP* sp\_userid = get\_encrypt\_obj(sp\_id).decryptAES(userid)  
  
 error = None  
 **if** request.method == **'POST'**:  
 *# get username & password hashes from html form* username = request.form[**'username'**]  
 password = hashlib.sha1(request.form[**'password'**]).hexdigest()  
 confirm\_pass = hashlib.sha1(request.form[**'confirmPassword'**]).hexdigest()  
  
 **if** confirm\_pass == password:  
 *# add this user to database* data\_base\_manager = DataBaseManager(DB\_NAME)  
 user\_id = data\_base\_manager.last\_id(**"Users"**) + 1  
 data\_base\_manager.close\_connection()  
 add\_user(user\_id, username, password, sp\_id, sp\_userid)  
  
 *# redirect to SP to continue registration* sp\_userid = get\_encrypt\_obj(sp\_id).encryptAES(sp\_userid)  
 addr = get\_sp\_url(sp\_id)  
 path = addr + **'/registeredas/'** + sp\_userid  
 server = **"PhoneBook Server"  
 return** render\_template(**'RedirectPage.html'**, path=path, server=server, addr=addr)  
 *#return redirect()* **else**:  
 *# if password confirmation was incorrect* error = **'Invalid Credentials. Please try again.'  
 return** render\_template(**'Register.html'**, error=error)  
*# endregion  
  
  
# region ------------------ OTHER\_FUNCTIONS ---------------------  
# get SP url from db (for redirect)***def** get\_sp\_url(sp\_id=1):  
 *"""  
 Gets RedirectPath data from SPs table in db, for a specific SP* **:param** *sp\_id: SP to find url of* **:return***: path of this SP  
 """* query = **"SELECT redirectPath FROM SPs WHERE SPID=%d"** % sp\_id  
 **try**:  
 data\_base\_manager = DataBaseManager(DB\_NAME)  
 path = data\_base\_manager.exec\_query(query)[0]  
 data\_base\_manager.close\_connection()  
 **return** path  
  
 **except** Exception, e:  
 **print 'Unable to execute query: '** + query  
 **print** e  
  
  
*# get encryption object for specific SP***def** get\_encrypt\_obj(sid):  
 *"""  
 Gets encryption key from SPs table in db.  
 Creates an instance of Encryption using that key.* **:param** *sid: SP server ID* **:return***: AES encryption object  
 """* query = **"SELECT key FROM SPs WHERE SPID=%d"** % sid  
 data\_base\_manager = DataBaseManager(DB\_NAME)  
 key = data\_base\_manager.exec\_query(query)[0]  
 data\_base\_manager.close\_connection()  
 key = Encryption(MASTER).decryptAES(key)  
 **return** Encryption(key)  
  
  
*# add user to UserProfiles table in db***def** add\_user(id, username, passw, sp\_id, sp\_user\_id):  
 *"""  
 Adds A row in UserProfiles table for a new user with this info:* **:param** *id: user ID* **:param** *username: chosen username* **:param** *passw: chosen password* **:param** *sp\_id: ID of SP server to which the user is registrating* **:param** *sp\_user\_id: ID o this user in SP server* **:return***: None  
 """* fields = [(id, username, passw, sp\_id, sp\_user\_id)]  
 data\_base\_manager = DataBaseManager(DB\_NAME)  
 data\_base\_manager.insert(**"Users"**, fields)  
 data\_base\_manager.print\_table(**"Users"**)  
 data\_base\_manager.close\_connection()  
  
  
*# create tables for DB***def** create\_db():  
 *# create Users table* data\_base\_manager = DataBaseManager(DB\_NAME)  
 fields = [**"ID integer primary key autoincrement"**, **"username text not null"**, **"passw text not null"**,  
 **"spID integer not null"**, **"spUserID integer not null"**]  
 data\_base\_manager.create\_table(**"Users"**, fields)  
  
 *# create SPs table* fields = [**"SPID integer primary key autoincrement"**, **"details text not null"**, **"redirectPath text not null"**,  
 **"key text not null"**]  
 data\_base\_manager.create\_table(**"SPs"**, fields)  
 data\_base\_manager.close\_connection()  
  
  
*# create a random key***def** generate\_random(size=16):  
 *"""  
 Generates random string 16 characters long.  
 (consists of uppercase and lowercase letters and numbers)* **:param** *size: length of string to generate* **:return***: 16 chars long random key  
 """* chars = string.ascii\_uppercase + string.ascii\_lowercase + string.digits  
 key = **''  
 for** i **in** range(size):  
 key += random.choice(chars)  
 **return** key  
  
  
*# socket communication with new SPs***def** sps\_comm():  
 *"""  
 Server side of socket comm.  
 Inserts newly connected SPs to DB.* **:return***: None  
 """* **while** True:  
 sp\_sock, sp\_addr = server\_sock.accept()  
 sp\_thread = ThreadWithReturnValue(target=register\_sp, args=(sp\_sock,))  
 sp\_thread.start()  
 fields = sp\_thread.join()  
 data\_base\_manager = DataBaseManager(**"MainDB.db"**)  
 data\_base\_manager.insert(**"SPs"**, fields)  
 data\_base\_manager.print\_table(**"SPs"**)  
 data\_base\_manager.close\_connection()  
  
  
*# exchange details to register new SP, with socket***def** register\_sp(sp\_sock):  
 *"""  
 Exchange of details between servers:  
 Sends encrypted key, ID for the new SP and receives SP address.* **:param** *sp\_sock:* **:return***:  
 """* **try**:  
 sp\_sock.send(**'OK Send name redirectpath'**)  
 **print 'OK Send name redirectpath'** *# receive SP address* sp\_info = sp\_sock.recv(1024)  
 **print** sp\_info  
 sp\_name, sp\_server\_path = sp\_info.split(**'@'**)  
  
 *# generate new key an sp ID* new\_id = 1  
 key = generate\_random()  
 *# change encryption to Asymmetric* key2send = Encryption(**b'Sixteen Byte Key'**).encryptAES(key)  
  
 *# send key and ID* **print** key2send + **'@'** + str(new\_id)  
 sp\_sock.send(key2send + **'@'** + str(new\_id))  
  
 ans = sp\_sock.recv(1024)  
 **print** ans  
 **if not** ans.startswith(**'OK'**):  
 **raise** ValueError  
 sp\_sock.close()  
  
 key = Encryption(MASTER).encryptAES(key)  
 **return** [(new\_id, sp\_name, sp\_server\_path, key), ]  
  
 **except** Exception, e:  
 **print "ERROR"  
 print** e  
  
  
*# Insert a few rows to Users table in db***def** insert\_init\_values():  
 data\_base\_manager = DataBaseManager(DB\_NAME)  
 fields = [(1, **'elle'**, hashlib.sha1(**'EL'**).hexdigest(), 1, 101), (2, **'David'**, hashlib.sha1(**'2511'**).hexdigest(), 1, 102),  
 (3, **'dana123'**, hashlib.sha1(**'12345'**).hexdigest(), 1, 103), (4, **'dana123'**, hashlib.sha1(**'12345'**).hexdigest(), 2, 101),  
 (5, **'elle'**, hashlib.sha1(**'EL'**).hexdigest(), 2, 102)]  
 data\_base\_manager.insert(**"Users"**, fields)  
  
 *# print db tables* data\_base\_manager.print\_table(**"Users"**)  
 data\_base\_manager.print\_table(**"SPs"**)  
  
 data\_base\_manager.close\_connection()  
*# endregion  
  
  
# region ------------------------ MAIN --------------------------***def** main():  
 create\_db()  
 insert\_init\_values()  
  
 *# create server socket listener (communication itself in new thread)* server\_sock.bind((HOST, int(PORT)+1))  
 server\_sock.listen(5)  
 thread.start\_new\_thread(sps\_comm, ())  
  
 app.run(host=HOST, port=PORT)  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()  
*# endregion*

**MySPWEB:**

*# region ----------------------- ABOUT --------------------------  
"""  
################################################################  
# Created By: Elizabeth Langerman #  
# Date: 04/2016 #  
# Name: Srvice Provider (SP) Server for example #  
# Version: 1.0 #  
# Windows Tested Versions: Win 8 32-bit, Win 7 32-bit #  
# Python Tested Versions: 2.7 32-bit #  
# Python Environment: PyCharm #  
################################################################  
"""  
# endregion  
  
# region ---------------------- IMPORTS -------------------------***from** DataBaseManager **import** \*  
**from** Encryption **import** \*  
**from** flask **import** Flask, request, redirect, render\_template  
**import** ConfigParser  
**import** socket  
**import** sys  
*# endregion  
  
# region ------------------ CONFIGURATIONS ----------------------*config = ConfigParser.ConfigParser()  
config.readfp(open(**'sp\_config.cfg'**)) *# path to config file***try**:  
 *# get constants from configurations file* MY\_IP = config.get(**'General'**, **'my\_ip'**)  
 PORT = int(config.get(**'General'**, **'port'**))  
 MAIN\_SERVER\_IP = config.get(**'General'**, **'MainServerIP'**)  
 MAIN\_SERVER\_PORT = int(config.get(**'General'**, **'MainServerPort'**))  
 DB\_NAME = config.get(**'General'**, **'db\_name'**)  
**except** (ConfigParser.NoOptionError, ConfigParser.NoSectionError), e:  
 **print** e  
 sys.exit(1)  
  
HOST = **'0.0.0.0'**MAIN\_SERVER\_PATH = **'http://'** + MAIN\_SERVER\_IP + **':'** + str(MAIN\_SERVER\_PORT)  
ME = **'PhoneBook'** + **'@'** + **'http://'** + MY\_IP + **':'** + str(PORT)  
*# endregion  
  
# region ----------------------- GLOBAL -------------------------*app = Flask(\_\_name\_\_)  
data\_base\_manager = DataBaseManager(DB\_NAME)  
*# endregion  
  
  
# region ------------------- URL\_FUNCTIONS ----------------------  
# route for SP index page*@app.route(**'/'**)  
**def** index():  
 **return** render\_template(**'PhoneBookIndex.html'**)  
  
  
*# route for SP login page*@app.route(**'/login'**, methods=[**'GET'**, **'POST'**])  
**def** login():  
 error = None  
 **if** request.method == **'POST'**:  
 username = request.form[**'username'**]  
 password = request.form[**'password'**]  
 **if** username != **'admin' or** password != **'admin'**:  
 error = **'Invalid Credentials. Please try again.'  
 else**:  
 **return** redirect(**'/login/%s %s'** % (username, password))  
 **return** render\_template(**'LoginSP.html'**, error=error)  
  
  
*# route for login page in option with main server*@app.route(**'/login/<server>'**)  
**def** private\_login(server):  
 *# redirect to My Main Server's login page* **if** server.lower() == **"mymainserver"**:  
 path = MAIN\_SERVER\_PATH + **'/login/'** + str(my\_sp\_id)  
 server = **"MyMainServer"  
 return** render\_template(**'RedirectPage.html'**, path=path, server=server, addr=MAIN\_SERVER\_PATH)  
 *#return redirect(MAIN\_SERVER\_PATH + '/login/' + str(my\_sp\_id))  
  
  
# route for showing the user's profile page*@app.route(**'/user/<userid>'**)  
**def** profile(userid):  
 *# show user profile* **try**:  
 userid = enc\_obj.decryptAES(userid)  
 **except** Exception, e:  
 **print** e  
 query = **"SELECT name, age, phoneNum FROM UserProfiles WHERE ID=%s"** % userid  
 name, age, phone\_num = data\_base\_manager.exec\_query(query)  
 **if** name:  
 **return** render\_template(**'Profile.html'**, name=name, age=age, phoneNum=phone\_num)  
 **else**:  
 *# error if userID was invalid* **print "Invalid User ID"***# route for creating new user ID*@app.route(**'/register'**)  
**def** register():  
 user\_id = data\_base\_manager.last\_id(**"UserProfiles"**) + 1  
 user\_id = str(user\_id)  
 user\_id = enc\_obj.encryptAES(user\_id)  
 *# redirect to My Main Server's registration page* path = (MAIN\_SERVER\_PATH+**"/register/"**+user\_id + **" "** + str(my\_sp\_id))  
 server = **"MyMainServer"  
 return** render\_template(**'RedirectPage.html'**, path=path, server=server, addr=MAIN\_SERVER\_PATH)  
 *#return redirect(MAIN\_SERVER\_PATH+"/register/"+user\_id + " " + str(my\_sp\_id))  
  
  
# route for SP registration page*@app.route(**'/registeredas/<user>'**, methods=[**'GET'**, **'POST'**])  
**def** registered(user):  
 user = enc\_obj.decryptAES(user)  
 error = None  
 **if** request.method == **'POST'**:  
 *# add this user to database* name = request.form[**'name'**]  
 age = request.form[**'age'**]  
 phone\_num = request.form[**'phoneNum'**]  
  
 **try**:  
 add\_user(user, name, age, phone\_num)  
 user = enc\_obj.encryptAES(user)  
 **return** redirect(**'/user/'**+user)  
 **except**:  
 error = **'Invalid Info. Please try again.'  
 return** render\_template(**'RegisterToSP.html'**, error=error)  
*# endregion  
  
  
# region ------------------ OTHER\_FUNCTIONS ---------------------  
# add user to UserProfiles table in db***def** add\_user(id, name, age, phone\_num):  
 *"""  
 Adds A row in UserProfiles table for a new user with this info:* **:param** *id: user ID* **:param** *name: name of user* **:param** *age: age of user* **:param** *phone\_num: his phone number* **:return***: None  
 """* fields = [(id, name, age, phone\_num)]  
 data\_base\_manager.insert(**"UserProfiles"**, fields)  
 *# data\_base\_manager.print\_table("UserProfiles")  
  
  
# create tables for DB***def** create\_db():  
 fields = [**"ID integer primary key autoincrement"**, **"name text not null"**, **"age integer not null"**,  
 **"phoneNum text not null"**]  
 data\_base\_manager.create\_table(**"UserProfiles"**, fields)  
 fields = [(101, **'Elizabeth'**, 17, **'09-8656735'**), (102, **'David'**, 10, **'03-1234567'**),  
 (103, **'Dana'**, 25, **'04-9182734'**)]  
 data\_base\_manager.insert(**"UserProfiles"**, fields)  
 *# data\_base\_manager.print\_table("UserProfiles")  
  
  
# get AES encryption object and sp ID***def** save\_key\_id(key, my\_id):  
 *"""* **:param** *key: the key to use for encryption* **:param** *my\_id: sp ID to save* **:return***: AES encryption object, my SP ID  
 """* key = Encryption(**b'Sixteen Byte Key'**).decryptAES(key)  
 enc = Encryption(key) *# correct enc\_obj: use to encrypt & decrypt sent data* my\_id = int(my\_id) *# send it for sp identification in redirect* **return** enc, my\_id  
  
  
*# connect to MyMainServer socket as client***def** connect\_mymainserver():  
 *"""  
 Creates a socket and connects to MyMainServer.  
 Exchange of details between servers:  
 Sends my address and receives encrypted key, ID.* **:return***: AES encryption object, my SP ID  
 """* my\_sock = socket.socket()  
 **print** MAIN\_SERVER\_IP, MAIN\_SERVER\_PORT  
 my\_sock.connect((MAIN\_SERVER\_IP, MAIN\_SERVER\_PORT+1))  
 ans = my\_sock.recv(1024)  
 **print** ans  
 **try**:  
 **if not** ans.startswith(**'OK'**):  
 **raise** ValueError  
  
 *# send my address* **print** ME  
 my\_sock.send(ME)  
  
 *# receive key, ID* data = my\_sock.recv(1024)  
 key, my\_id = data.split(**'@'**)  
 my\_sock.send(**'OK End Connection'**)  
 my\_sock.close()  
  
 **return** save\_key\_id(key, my\_id)  
  
 **except** Exception:  
 my\_sock.send(**'Error'**)  
 my\_sock.close()  
 *#return connect\_mymainserver()  
# endregion  
  
  
# region ------------------------ MAIN --------------------------*create\_db()  
enc\_obj, my\_sp\_id = connect\_mymainserver()  
  
  
**def** main():  
 app.run(host=HOST, port=PORT) *# run Flask app (Web server)* data\_base\_manager.close\_connection()  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()  
*# endregion*

**DataBaseManager:**

**import** sqlite3  
  
  
**class** DataBaseManager(object):  
  
 *# constructor* **def** \_\_init\_\_(self, name):  
 self.con = sqlite3.connect(name, )  
 self.cur = self.con.cursor()  
  
 *# create a table in db* **def** create\_table(self, table\_name, fields):  
 *"""* **:param** *table\_name: name of new table to create* **:param** *fields: structure of fields for that table* **:return***: builds a suitable query and executes it  
 """* query = **"CREATE TABLE IF NOT EXISTS "** + table\_name + **"("  
 for** field **in** fields:  
 query += field + **','** *# remove the last ',' and close with ')'* query = query[:-1]  
 query += **")"** self.cur.execute(query)  
  
 *# executes a given query* **def** exec\_query(self, query):  
 *"""* **:param** *query: query to execute* **:return***: query return values  
 """* self.cur.execute(query)  
 **return** self.cur.fetchone()  
  
 *# insert values to table in db* **def** insert(self, table\_name, fields):  
 *"""* **:param** *table\_name: the table to insert into* **:param** *fields: values to be inserted* **:return***: builds a suitable query and executes it  
 """* query = **"INSERT INTO "** + table\_name + **" VALUES("  
 for** i **in** xrange(len(fields[0])):  
 query += **'?,'** *# remove the last ',' and close with ')'* query = query[:-1]  
 query += **")"** self.cur.executemany(query, fields)  
  
 *# get last row num(ID)* **def** last\_id(self, table\_name):  
 *"""* **:param** *table\_name: the table to check* **:return***: id of last row in table  
 """* rows = self.get\_rows(table\_name)  
 **return** int(rows[-1][0])  
  
 *# get all rows of a table* **def** get\_rows(self, table\_name):  
 *"""* **:param** *table\_name: the table to get rows frm* **:return***: builds a suitable query and executes it  
 """* self.cur.execute(**"SELECT \* FROM "** + table\_name)  
 rows = self.cur.fetchall()  
 **return** rows  
  
 *# print all table values* **def** print\_table(self, table\_name):  
 *"""* **:param** *table\_name: table to print* **:return***: None  
 """* rows = self.get\_rows(table\_name)  
 **for** row **in** rows:  
 **print** row  
  
 *# close connection with db file* **def** close\_connection(self):  
 self.con.commit()  
 self.con.close()

**ThreadWithReturnValue:**

**from** threading **import** Thread  
  
  
**class** ThreadWithReturnValue(Thread):  
  
 *# constructor* **def** \_\_init\_\_(self, group=None, target=None, name=None, args=(), kwargs={}, Verbose=None):  
 Thread.\_\_init\_\_(self, group, target, name, args, kwargs, Verbose)  
 self.\_return = None  
  
 *# the main thread function* **def** run(self):  
 *"""  
 Run thread and save its return value* **:return***: None  
 """* **if** self.\_Thread\_\_target **is not** None:  
 self.\_return = self.\_Thread\_\_target(\*self.\_Thread\_\_args, \*\*self.\_Thread\_\_kwargs)  
  
 *# overrides join function in threading.Thread* **def** join(self):  
 *"""  
 Returns the value saved during run function (self.\_return)* **:return***: Thread return value  
 """* Thread.join(self)  
 **return** self.\_return

**Encryption:**

**from** Crypto.Cipher **import** AES  
**import** base64  
  
  
**class** Encryption(object):  
  
 *# constructor* **def** \_\_init\_\_(self, key, enc\_type=**"AES"**, block=16):  
 self.\_\_cipher = AES.new(key)  
 self.encrypt\_type = enc\_type  
 self.block\_size = block  
 self.padding = **'{'** *# add padding to message to get right block size* **def** add\_padding(self, data):  
 *"""* **:param** *data: message to pad* **:return***: padded data according to self.block\_size  
 """* num = self.block\_size - len(data) % self.block\_size  
 **return** data + num \* self.padding  
  
 *# encryption function* **def** encryptAES(self, data):  
 *"""  
 Adds padding to data, encrypts it with AES, and encodes with base64  
 (also deals with a case of '/' in string)* **:param** *data: message to encrypt* **:return***: padded, encrypted, encoded data  
 """* padded = self.add\_padding(data)  
 encrypted = self.\_\_cipher.encrypt(padded)  
 encoded = base64.b64encode(encrypted)  
 final = self.remove\_slash(encoded)  
 **return** final  
  
 *# decryption function* **def** decryptAES(self, encoded):  
 *"""  
 Decodes message with base64, decrypts with AES, and removes padding  
 (also deals with a case of '/' in string)* **:param** *encoded: message to encrypt* **:return***: the original message  
 """* correct = self.add\_slash\_again(encoded)  
 decoded = base64.b64decode(correct)  
 decrypted = self.\_\_cipher.decrypt(decoded)  
 no\_padding = decrypted.rstrip(self.padding)  
 **return** no\_padding  
  
 *# changes all '/' to '$'* **def** remove\_slash(self, data):  
 *"""  
 If '/' is in the encrypted string, it is viewed as part of the URL,  
 which causes a problem in defining the correct page path.  
 This function solves the problem by changing all '/' to '$'.* **:param** *data: encrypted message* **:return***: data with all '/' changed to '$'  
 """* **while '/' in** data:  
 i = data.find(**'/'**)  
 data = data[:i] + **'$'** + data[i+1:]  
 **return** data  
  
 *# changes all '$' back to '/'* **def** add\_slash\_again(self, data):  
 *"""  
 To properly decrypt a message, all '$' should be changed bac to '/'.  
 Otherwise decryption won't return the original message.* **:param** *data: encrypted message* **:return***: data with all '$' changed back to '/'  
 """* **while '$' in** data:  
 i = data.find(**'$'**)  
 data = data[:i] + **'/'** + data[i+1:]  
 **return** data

**LoginMain:**

<html>

<head>

<title>Login page</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="http://icons.iconarchive.com/icons/social-media-icons/glossy-social/512/Spring-me-icon.png" rel="shortcut icon">

</head>

<body>

<div class="container">

<center>

<h1>Please login</h1>

<br>

<form action="" method="post">

<input type="text" placeholder="Username" name="username" ><br/><br/><br/>

<input type="password" placeholder="Password" name="password" ><br/><br/><br/>

<input class="btn btn-default" type="submit" value="Login">

</form>

<!-- <form method="get" action="/register">

<input class="btn btn-default" type="submit" value="Register">

</form> -->

{% if error %}

<p class="error"><strong>Error:</strong> {{ error }}

{% endif %}

</center>

</div>

</body>

</html>

**Register: (Main)**

<html>

<head>

<title>Registration page</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="http://icons.iconarchive.com/icons/social-media-icons/glossy-social/512/Spring-me-icon.png" rel="shortcut icon">

</head>

<body>

<div class="container">

<center>

<h1>Please enter credentials</h1>

<br>

<form action="" method="post">

<input type="text" placeholder="Username" name="username" ><br/><br/><br/>

<input type="password" placeholder="Password" name="password" ><br/><br/><br/>

<input type="password" placeholder="Confirm Password" name="confirmPassword" ><br/><br/><br/>

<input class="btn btn-default" type="submit" value="Register">

</form>

{% if error %}

<p class="error"><strong>Error:</strong> {{ error }}

{% endif %}

</center>

</div>

</body>

</html>

**PhoneBookIndex:**

<html>

<head>

<title>Welcome To PhoneBook</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="http://www.isneo.org/wp-content/uploads/phonebook.png" rel="shortcut icon">

</head>

<body>

<center>

<h1>Hello Dear Guest!</h1>

<h2><strong>Welcome To PhoneBook!</strong></h2>

<p>We want to help you save your phone number.<br/>

During registration you will be asked to enter your name, age & phone number.<br/>

When logged in, we will show you these.</big></p>

<p><big>You don't have to worry about keeping your phone number in memory anymore!<br/>

Just log in and see it:</big></p>

<form method="get" action="/login">

<input class="btn btn-default" type="submit" value="Login">

</form>

<form method="get" action="/register">

<input class="btn btn-default" type="submit" value="Register">

</form>

<br/>

<img src="http://thumbs.dreamstime.com/t/phone-book-concept-public-35018949.jpg">

</center>

</body>

</html>

**LoginSP**

<html>

<head>

<title>Login page</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="http://www.isneo.org/wp-content/uploads/phonebook.png" rel="shortcut icon">

</head>

<body>

<div class="container">

<center>

<h1>Please login</h1>

<br/>

<form action="" method="post">

<input type="text" placeholder="Username" name="username" ><br/><br/>

<input type="password" placeholder="Password" name="password" ><br/>

</form>

<form method="get" action="/login">

<input class="btn btn-default" type="submit" value="Login with username & password">

</form>

or<br/><br/>

<form method="get" action="/login/mymainserver">

<input class="btn btn-default" type="submit" value="Login with MyMainServer (preferable)">

</form>

{% if error %}

<p class="error"><strong>Error:</strong> {{ error }}

{% endif %}

</center>

</div>

</body>

</html>

**RegisterToSP:**

<html>

<head>

<title>Registration page</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="http://www.isneo.org/wp-content/uploads/phonebook.png" rel="shortcut icon">

</head>

<body>

<div class="container">

<center>

<h1>Please enter your information</h1>

<br>

<form action="" method="post">

<input type="text" placeholder="Name" name="name" ><br/><br/><br/>

<input type="number" placeholder="Age" name="age" min="1" max="120"><br/><br/><br/>

<input type="text" placeholder="Phone Number" name="phoneNum" ><br/><br/><br/>

<input class="btn btn-default" type="submit" value="Register">

</form>

{% if error %}

<p class="error"><strong>Error:</strong> {{ error }}

{% endif %}

</center>

</div>

</body>

</html>

**Profile:**

<html>

<head>

<title>Your Profile</title>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="http://www.isneo.org/wp-content/uploads/phonebook.png" rel="shortcut icon">

</head>

<body>

<center>

{% if name %}

<h1>Hello {{ name }}</h1>

<br/><br/><br/>

<p><big>Your name is: </big></p>

<big><strong> {{ name }} </strong></big><br/><br/><br/>

<p><big>Your age is:</big></p>

<big><strong> {{ age }} </strong></big><br/><br/><br/>

<p><big>Your phone number is: </big></p>

<big><strong> {{ phoneNum }} </strong></big><br/><br/><br/>

{% endif %}

</center>

</body>

</html>

**RedirectPage:**

<html>

<head>

<META HTTP-EQUIV="refresh" CONTENT="4;URL={{ path }}">

</head>

<body>

<h2>Redirecting to {{ server }}</h2>

<h2>{{ addr }}</h2>

</body>

</html>