|  |  |
| --- | --- |
|  | Student Council Election  MERCADO, Luis Benjamin Z.  SIOSON, Glenn Christian D.  TAN, Lauren C. |

1. Table of Contents

[A Objectives 1](#_Toc527330005)

[B Scope and Features 2](#_Toc527330006)

[B.1 Scope and Feature # 1 2](#_Toc527330007)

[B.2 Scope and Feature # 2 2](#_Toc527330008)

[B.3 Scope and Feature # 3 2](#_Toc527330009)

[B.4 Scope and Feature # 4 2](#_Toc527330010)

[C Limitations 3](#_Toc527330011)

[C.1 Limitation # 1 3](#_Toc527330012)

[C.2 Limitation # 2 3](#_Toc527330013)

[D System Requirements 4](#_Toc527330014)

[D.1 Hardware Requirements 4](#_Toc527330015)

[D.1.1 Processor 4](#_Toc527330016)

[D.1.2 Memory 4](#_Toc527330017)

[D.2 Software Requirements 4](#_Toc527330018)

[D.2.1 Operating System 4](#_Toc527330019)

[D.2.2 Software Requirements 4](#_Toc527330020)

[E Class Diagram 5](#_Toc527330021)

[E.1 Student Council Election Class Diagram 5](#_Toc527330022)

[F Entity Relationship Diagram 6](#_Toc527330023)

[F.1 Student Council Election Entity Relationship Diagram 6](#_Toc527330024)

[G Class Definition 7](#_Toc527330025)

[H User Manual 15](#_Toc527330026)

[H.1 Login Page 15](#_Toc527330027)

[H.2 Election Panel 16](#_Toc527330028)

[H.3 Create User (Admin) 17](#_Toc527330029)

[H.4 Delete User (Admin) 18](#_Toc527330030)

[H.5 Edit User (Admin) 19](#_Toc527330031)

[H.6 Promote User to Candidate (Admin) 20](#_Toc527330032)

[H.7 See Summary of Votes (Admin) 21](#_Toc527330033)

[I References 22](#_Toc527330034)

[I.1 Reference # 1 22](#_Toc527330035)

[I.2 Reference # 2 22](#_Toc527330036)

List of Figures

[Figure F.1.a: {Name} Class Diagram 5](#_Toc515468271)

[Figure H.1.a: User Manual Image # 1 8](#_Toc515468272)

[Figure H.2.a: User Manual Image # 2 9](#_Toc515468273)

[Figure H.3.a: User Manual Image # 3 10](#_Toc515468274)

# Objectives

The objective of this project is to create a python application utilizing the 3-tier architecture. This architecture pattern includes the user interface (UI), functional process logic ("business Logic"), computer data storage and data access as independent modules (“Data Access Layer”).

The project should also include the usage of a database management system such as SQLite or MySQL that will be embedded into the end program.

The project should be executed in the PC’s terminal/command prompt, all files should be compiled in the master’s branch of the project repository in github. The files should include proper documentation, user manual, database, resources and the program files.

# Scope and Features

## Scope and Feature # 1

The program features a student council election system that uses SQLite as its database. The program allows the admin to start and end the election. The user could vote for the candidates only if the election is started by the admin. The admin can view the total number of votes per candidate. Once the election is ended by the admin, the results will be displayed.

## Scope and Feature # 2

Users can login using their MyMapua account, the default password for the account is their student number which can be later on be changed. Once logged in, the user will be redirected to the main window wherein the user could change their password, and vote for the candidate they desire, after voting the user can submit their vote ticket by clicking the submit vote button.

## Scope and Feature # 3

The voting window includes the list of positions and its respective candidate. Both candidates running for each position are displayed alongside their picture, name and their platform. If the user decides not to vote for a certain position, it will be recorded as abstain.

## Scope and Feature # 4

The admin account is preset, admins can start and end the elections, create users, update user information, add/remove user and promote a user to a candidate.

# Limitations

## Limitation # 1

The application is limited on using a local database which is the SQLite

## Limitation # 2

The users are only from the EECE department of Mapua.

# System Requirements

## Hardware Requirements

### Processor

Processor Core Count : 2 or above

### Memory

Memory Capacity : 256 MB RAM or above

Disk Size : 1 GB above

## Software Requirements

### Operating System

Operating System Name : Windows7/8/10, Linux/UNIX

Edition : Windows 7

Starter, Home Basic, Home Premium, Professional, Enterprise, Ultimate

Windows 8

Windows 8, Pro, Enterprise, RT

Windows 10

Enterprise, Pro, Education,

Linux

Ubuntu, Fedora, Mint

### Software Requirements

Software Application # 1 : Python 3.5

# Class Diagram

## Student Council Election Class Diagram

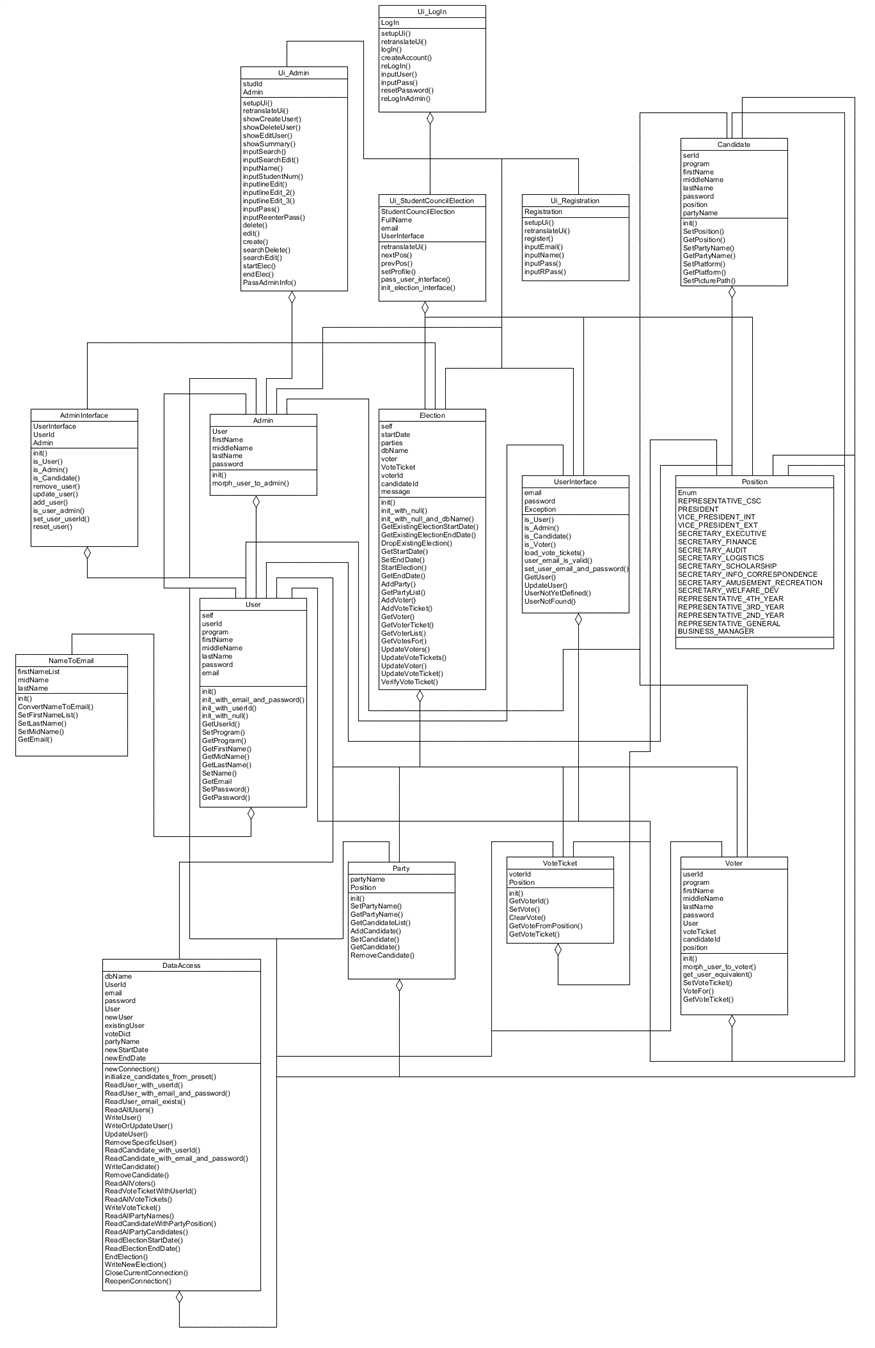


Figure F.1.a: Student Council Election Class Diagram

(Refer to Documentation/ClassDiagram.png or Class Diagram.uxf for a larger version)

# Entity Relationship Diagram

## Student Council Election Entity Relationship Diagram

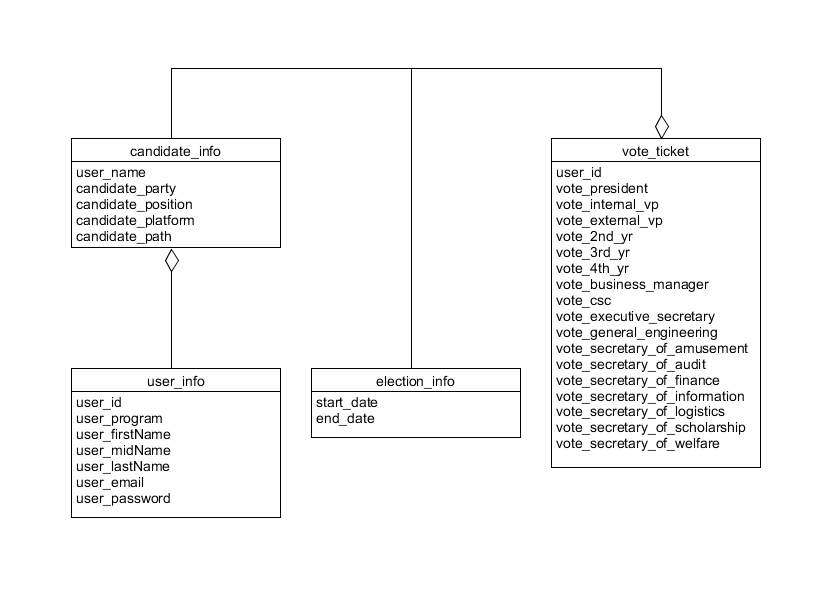


Figure F.1.b: Entity Relationship Diagram used for the Database

(Refer to Documentation/Entity Relationship Diagram.png or Class Entity Relationship Diagram.uxf for a larger version)

# Class Definition

|  |  |
| --- | --- |
| **Name:** | Ui\_LogIn |
| **Type:** | Class |
| **Description:** | Class for the Login UI |
| **Fields:** | LogIn |
| **Methods:** | setupUi(self, LogIn) |
| retranslateUi(self, LogIn) |
| logIn(self) |
| createAccount(self) |
| reLogIn(self) |
| inputUser(self) |
| inputPass(self) |
| resetPassword(self) |
| reLogInAdmin(self) |

|  |  |
| --- | --- |
| **Name:** | Ui\_Admin |
| **Type:** | Class |
| **Description:** | Class for the Admin UI |
| **Fields:** | studId  Admin |
| **Methods:** | setupUi() |
| retranslateUi() |
| showCreateUser() |
| showDeleteUser() |
| showEditUser() |
| showSummary() |
| inputSearch() |
| inputSearchEdit() |
| inputName() |
| inputStudentNum() |
| inputlineEdit() |
| inputlineEdit\_2() |
| inputlineEdit\_3() |
| inputPass() |
| inputReenterPass() |
| delete() |
| edit() |
| create() |
| searchDelete() |
| searchEdit() |
| startElec() |
| endElec() |
| PassAdminInfo() |
| **Name:** | Ui\_Registration |
| **Type:** | Class |
| **Description:** | Class for Registration UI |
| **Fields:** | Registration |
| **Methods:** | setupUi() |
| retranslateUi() |
| register() |
| inputEmail() |
| inputName() |
| inputPass() |
| inputRPass() |

|  |  |
| --- | --- |
| **Name:** | Ui\_StudentCouncilElection |
| **Type:** | Class |
| **Description:** | Class for the Student Council Election UI |
| **Fields:** | StudentCouncilElection  FullName  Email UserInterface |
| **Methods:** | retranslateUi() |
| nextPos() |
| prevPos() |
| setProfile() |
| pass\_user\_interface() |
| init\_election\_interface() |

|  |  |
| --- | --- |
| **Name:** | Admin |
| **Type:** | Class |
| **Description:** | Class for morphing user to admin |
| **Fields:** | User  firstName  middleName  lastName  password |
| **Methods:** | init() |
| morph\_user\_to\_admin() |

|  |  |
| --- | --- |
| **Name:** | AdminInterface |
| **Type:** | Class |
| **Description:** | Class for the Admin Interface |
| **Fields:** | UserInterface  UserId  Admin |
| **Methods:** | init() |
| is\_User() |
| is\_Admin() |
| is\_Candidate() |
| remove\_user() |
| update\_user() |
| add\_user() |
| is\_user\_admin() |
| set\_user\_userId() |
| reset\_user() |

|  |  |
| --- | --- |
| **Name:** | Candidate |
| **Type:** | Class |
| **Description:** | Class for the Candidates |
| **Fields:** | serId  program  firstName  middleName  lastName  password  position  partyName |
| **Methods:** | init() |
| SetPosition() |
| GetPosition() |
| SetPartyName() |
| GetPartyName() |
| SetPlatform() |
| GetPlatform() |
| SetPicturePath() |
| GetPicturePath() |

|  |  |
| --- | --- |
| **Name:** | Election |
| **Type:** | Class |
| **Description:** | Class for Election |
| **Fields:** | self  startDate  parties  dbName  voter  VoteTicket  voterId  candidateId  message |
| **Methods:** | init() |
| init\_with\_null() |
| init\_with\_null\_and\_dbName() |
| GetExistingElectionStartDate() |
| GetExistingElectionEndDate() |
| DropExistingElection() |
| GetStartDate() |
| SetEndDate() |
| StartElection() |
| GetEndDate() |
| AddParty() |
| GetPartyList() |
| AddVoter() |
| AddVoteTicket() |
| GetVoter() |
| GetVoterTicket() |
| GetVoterList() |
| GetVotesFor() |
| UpdateVoters() |
| UpdateVoteTickets() |
| UpdateVoter() |
| UpdateVoteTicket() |
| VerifyVoteTicket() |

|  |  |
| --- | --- |
| **Name:** | NameToEmail |
| **Type:** | Class |
| **Description:** | Class for the conversion of Name To Email |
| **Fields:** | firstNameList  midName  lastName |
| **Methods:** | init() |
| ConvertNameToEmail() |
| SetFirstNameList() |
| SetLastName() |
| SetMidName() |
| GetEmail() |
| **Name:** | Party |
| **Type:** | Class |
| **Description:** | Class for the Parties |
| **Fields:** | partyName  Position |
| **Methods:** | init() |
| SetPartyName() |
| GetPartyName() |
| GetCandidateList() |
| AddCandidate() |
| SetCandidate() |
| GetCandidate() |
| RemoveCandidate() |

|  |  |
| --- | --- |
| **Name:** | Position |
| **Type:** | Class |
| **Description:** | Class for the Positions |
| **Fields:** | Enum  REPRESENTATIVE\_CSC  PRESIDENT  VICE\_PRESIDENT\_INT  VICE\_PRESIDENT\_EXT  SECRETARY\_EXECUTIVE  SECRETARY\_FINANCE  SECRETARY\_AUDIT  SECRETARY\_LOGISTICS  SECRETARY\_SCHOLARSHIP  SECRETARY\_INFO\_CORRESPONDENCE  SECRETARY\_AMUSEMENT\_RECREATION  SECRETARY\_WELFARE\_DEV  REPRESENTATIVE\_4TH\_YEAR  REPRESENTATIVE\_3RD\_YEAR  REPRESENTATIVE\_2ND\_YEAR  REPRESENTATIVE\_GENERAL  BUSINESS\_MANAGER |

|  |  |
| --- | --- |
| **Name:** | User |
| **Type:** | Class |
| **Description:** | Class for the Users |
| **Fields:** | self  userId  program  firstName  middleName  lastName  password  email |
| **Methods:** | init() |
| init\_with\_email\_and\_password() |
| init\_with\_userId() |
| init\_with\_null() |
| GetUserId() |
| SetProgram() |
| GetProgram() |
| GetFirstName() |
| GetMidName() |
| GetLastName() |
| SetName() |
| GetEmail |
| SetPassword() |
| GetPassword() |

|  |  |
| --- | --- |
| **Name:** | UserInterface |
| **Type:** | Class |
| **Description:** | Class for the User Interface |
| **Fields:** | email  password  Exception |
| **Methods:** | is\_User() |
| is\_Admin() |
| is\_Candidate() |
| is\_Voter() |
| load\_vote\_tickets() |
| user\_email\_is\_valid() |
| set\_user\_email\_and\_password() |
| GetUser() |
| UpdateUser() |
| UserNotYetDefined() |
| UserNotFound() |

|  |  |
| --- | --- |
| **Name:** | Voter |
| **Type:** | Class |
| **Description:** | Class for the Voter |
| **Fields:** | userId  program  firstName  middleName  lastName  password  User  voteTicket  candidateId  position |
| **Methods:** | init() |
| morph\_user\_to\_voter() |
| get\_user\_equivalent() |
| SetVoteTicket() |
| VoteFor() |
| GetVoteTicket() |

|  |  |
| --- | --- |
| **Name:** | VoteTicket |
| **Type:** | Class |
| **Description:** | Class for the Vote Ticket |
| **Fields:** | voterId  Position |
| **Methods:** | init() |
| GetVoterId() |
| SetVote() |
| ClearVote() |
| GetVoteFromPosition() |
| GetVoteTicket() |

|  |  |
| --- | --- |
| **Name:** | DataAccess |
| **Type:** | Class |
| **Description:** | Class for the Data Access Layer |
| **Fields:** | dbName  UserId  email  password  User  newUser  existingUser  voteDict  partyName  newStartDate  newEndDate |
| **Methods:** | newConnection() |
| initialize\_candidates\_from\_preset() |
| ReadUser\_with\_userId() |
| ReadUser\_with\_email\_and\_password() |
| ReadUser\_email\_exists() |
| ReadAllUsers() |
| WriteUser() |
| WriteOrUpdateUser() |
| UpdateUser() |
| RemoveSpecificUser() |
| ReadCandidate\_with\_userId() |
| ReadCandidate\_with\_email\_and\_password() |
| WriteCandidate() |
| RemoveCandidate() |
| ReadAllVoters() |
| ReadVoteTicketWithUserId() |
| ReadAllVoteTickets() |
| WriteVoteTicket() |
| ReadAllPartyNames() |
| ReadCandidateWithPartyPosition() |
| ReadAllPartyCandidates() |
| ReadElectionStartDate() |
| ReadElectionEndDate() |
| EndElection() |
| WriteNewElection() |
| CloseCurrentConnection() |
| ReopenConnection() |

# User Manual

## Login Page

In order to start the program, the user must run the main.py located in Council Election\Codes

Once executed, the login screen will open

Fill up the email address alongside its password accordingly



Figure H.1.a: Login Page

## H.2 Election Panel

After logging in, if the user uses a student account, the user will be redirected in the election panel, if the user logs in as admin it will be redirected in the admin panel.

The user (voter) can only enter once the admin has started the election

If the election is started the user will be shown the positions alongside the candidates whom the user will choose from.

The arrows are used to navigate between the positions, the list of positions located at the left side can also be used.

For each position, the user will choose either of the two candidates, if the user did not choose anyone, the vote will be recorded as empty for the specific position.

Once the user is finished voting, the vote is ready to be submitted.

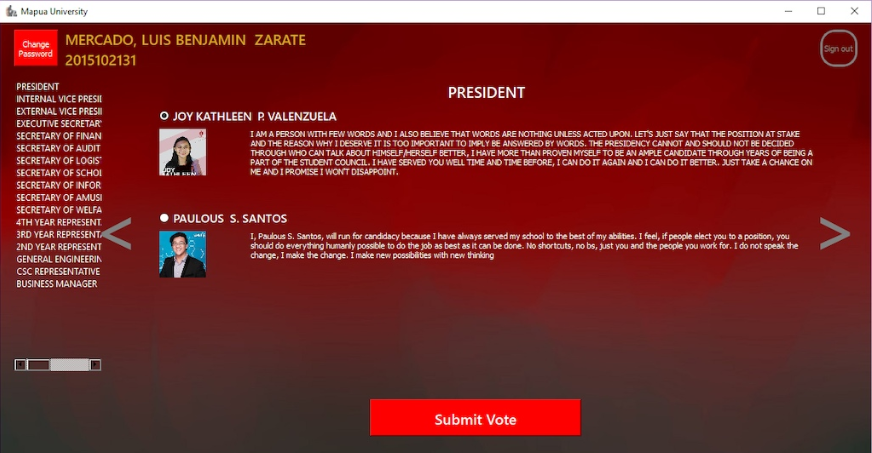


Figure H.2.a: Election Page

## Create User (Admin)

The user will only be directed to the admin page using an admin account.

After entering the admin page, the admin can choose from the following options namely Create User, Delete User, Edit User, Start Election and End Election.

If the admin chooses to start the election, it is the only time where the user can enter the election panel and vote for the candidates.

For the create user tab, the admin can add a user to the database by providing the necessary information in the given format.

After entering the information choose whether to add the user as a normal user or as an admin, once finished, click the Create button to finalize.

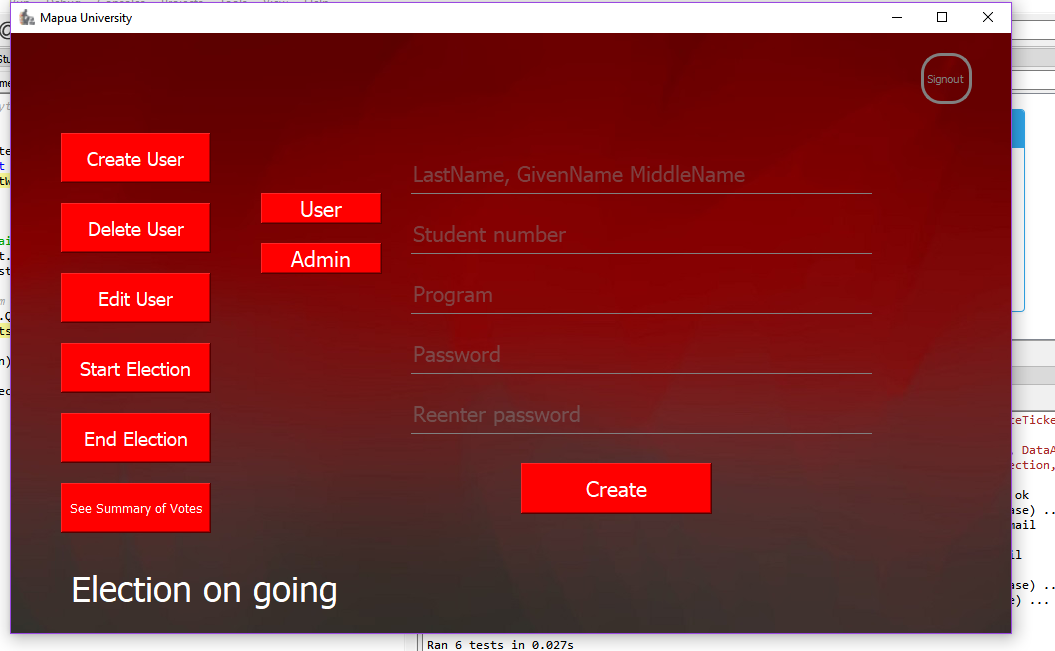


Figure H.3.a: Create User Tab

## Delete User (Admin)

The admin can delete a user provided their Student Number

After entering the student number, and clicking search, the information about the student will appear.

Click the delete button to delete the user from the database.

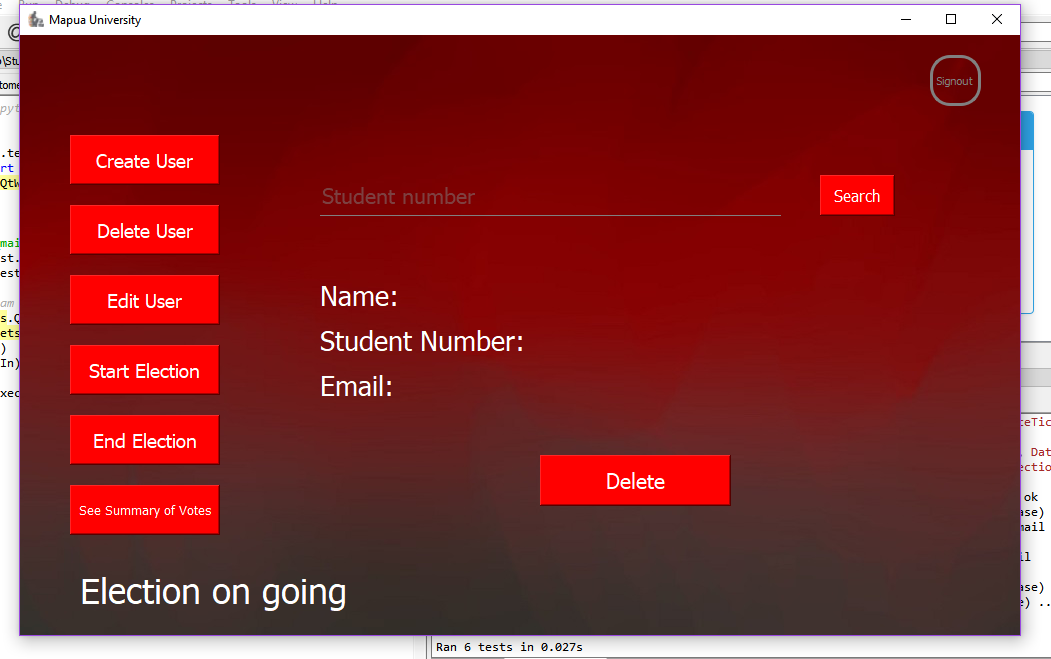


Figure H.4.a: Delete User Tab

## Edit User (Admin)

The admin can edit a user provided their Student Number

After entering the student number, and clicking search, the information about the student can be edited.

The student can either be promoted to a candidate if it is a user, or the candidate can be demoted to a user depending on what the admin will pick.

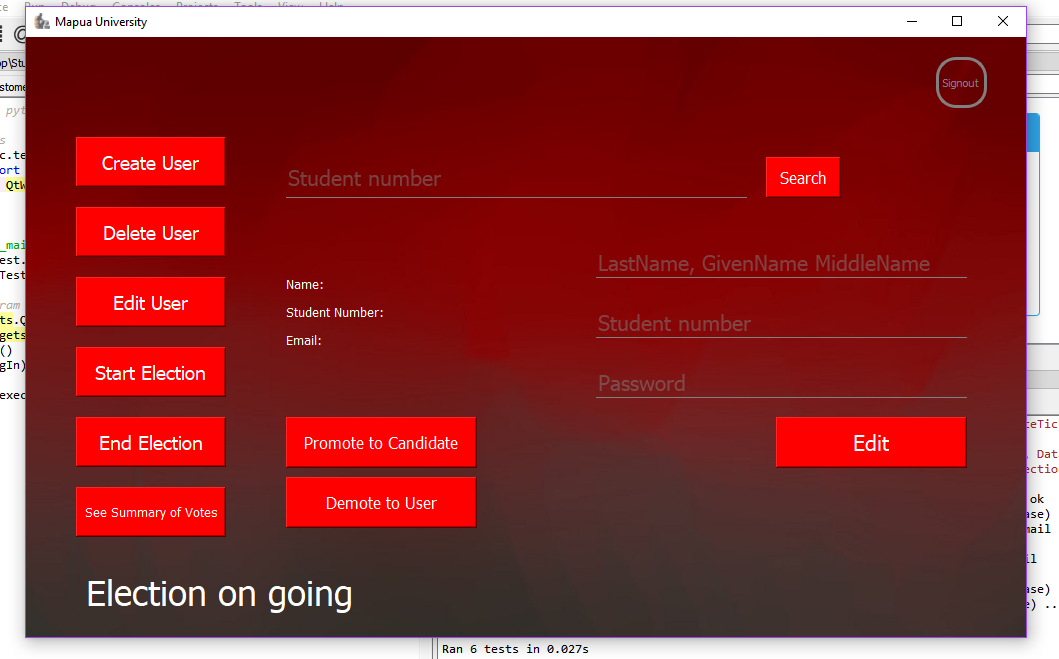


Figure H.5.a: Edit User Tab

## Promote User to Candidate (Admin)

While at the Edit User tab, if the admin decided to promote the user to a candidate, the admin will be shown fields to fill up.

The admin should fill up what party, position and platform the user will have. Then a copy for the path of the user’s 120px120p picture shall be provided.

After filling up all the fields, click on the promote button to finalize everything.

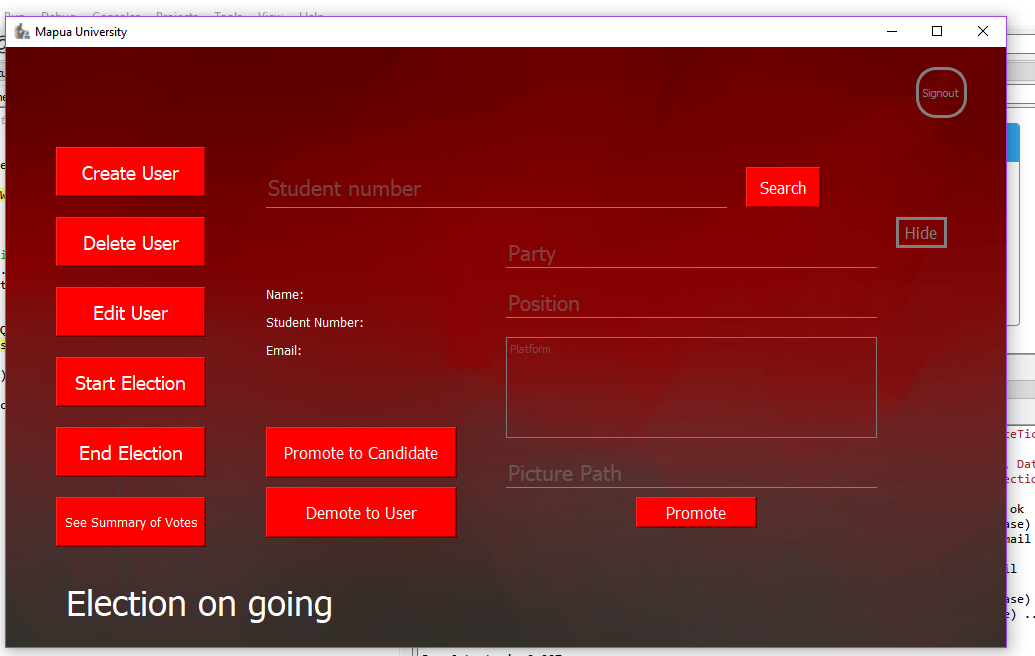


Figure H.6.a: Promoting User Tab

## See Summary of Votes (Admin)

If the Summary of votes are clicked, it will display a drop-down menu for all the positions present.

If a position is chosen, it will display the candidates as well as its total number of votes.

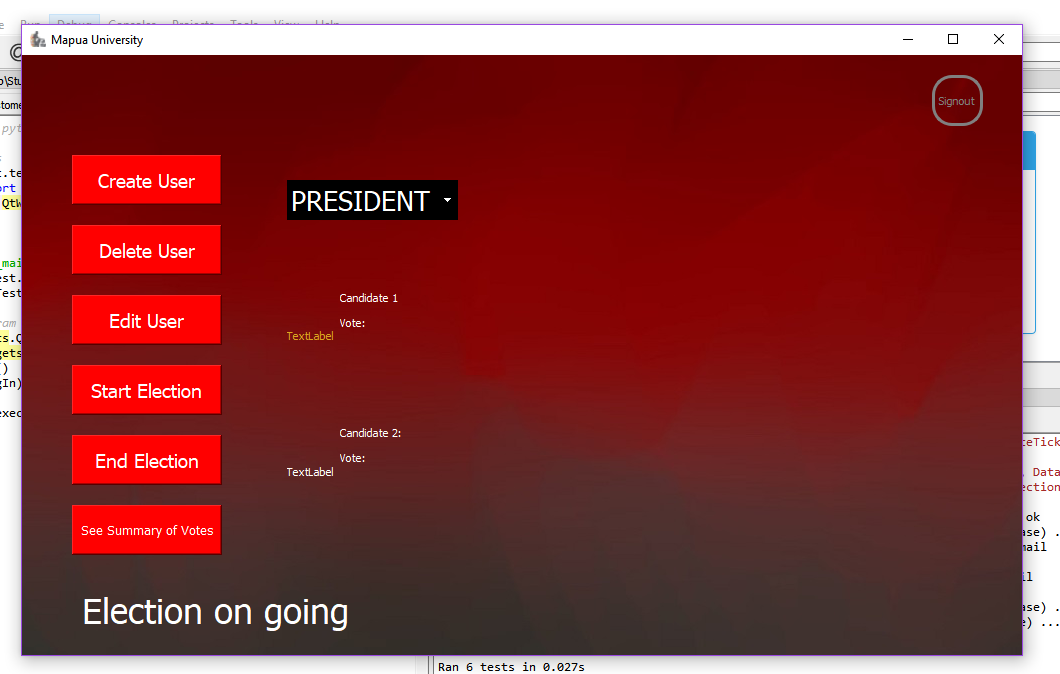


Figure H.7.a: Summary of Votes Tab

# References

## Reference # 1

Reference Title : Unit Test — Unit testing framework

Reference Author : Python Software Foundation

Reference Page Source : docs.python.org/2/library/unittest.html

## Reference # 2

Reference Title : Testing Your Code

Reference Author : Kenneth Reitz

Reference Page Source : docs.python-guide.org/writing/tests/

