**Task 6 Testing:**

***1 Testing Environment and Test Cases:***

Environment Test Framework:

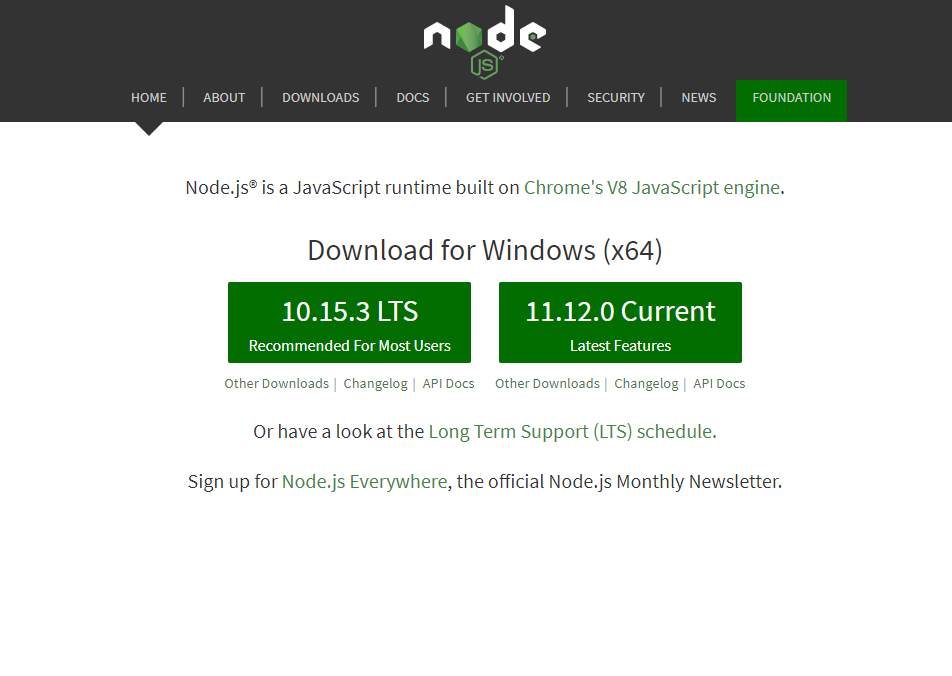
In this framework we decided to use:

* NodeJS for Javascript (frontend)
* SQLiteStudio for SQLite (backend)
* IntelliJ IDEA for Java (unit test)

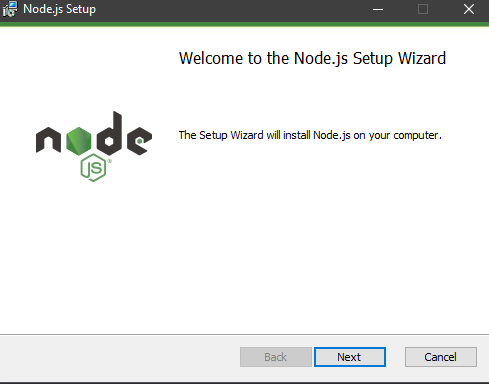
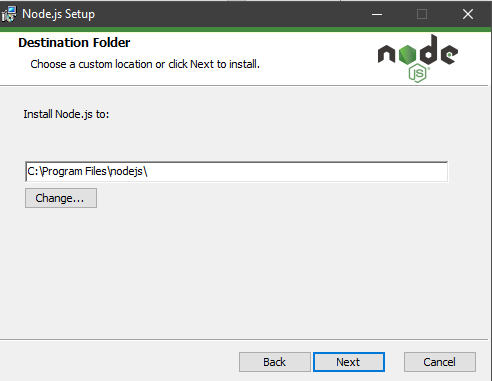
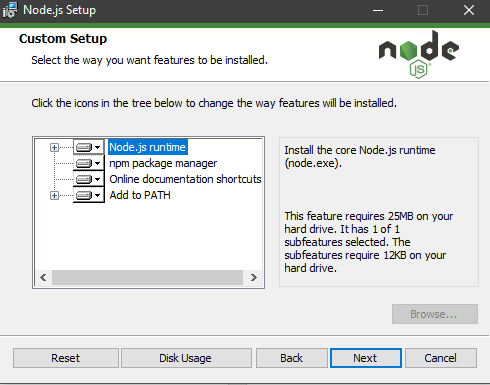
A: Installation of Framework:

*NodeJS:*

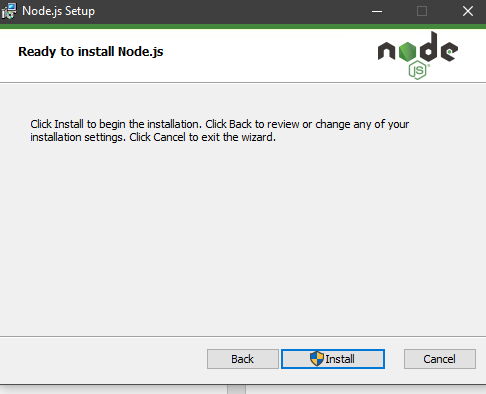
Step 1: Go to their website and click download current version for your chosen OS



Step 2: Follow the setup wizard form the downloaded file and accept prompts that pop up

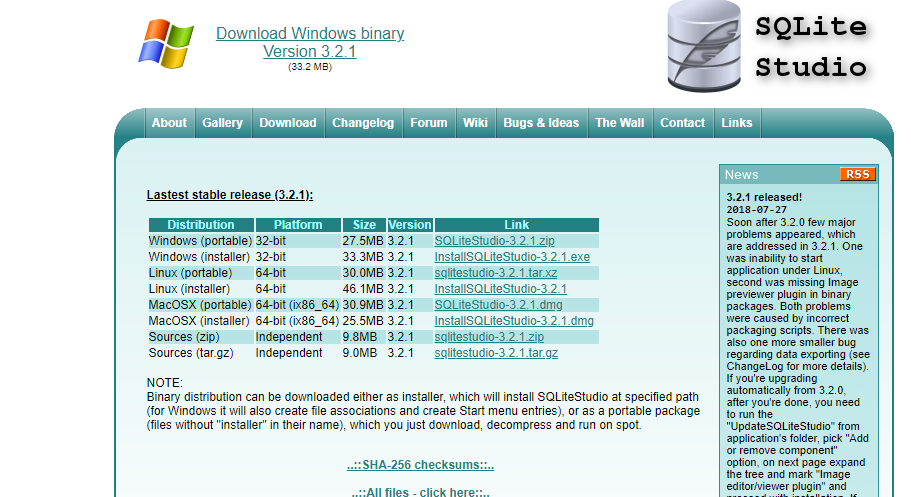


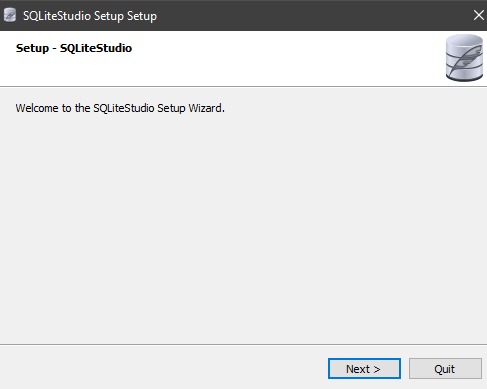
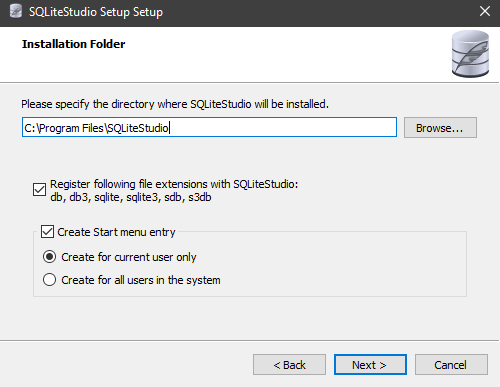
Step 3: Click install and now you can use the IDE for JavaScript

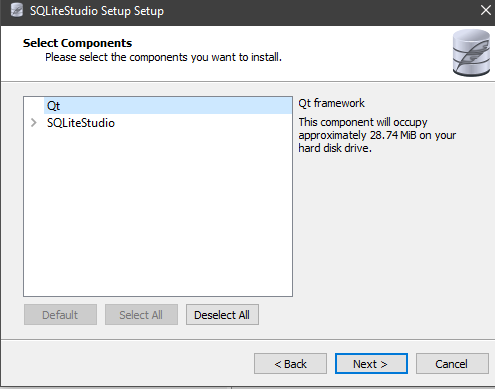


*SQLite:*

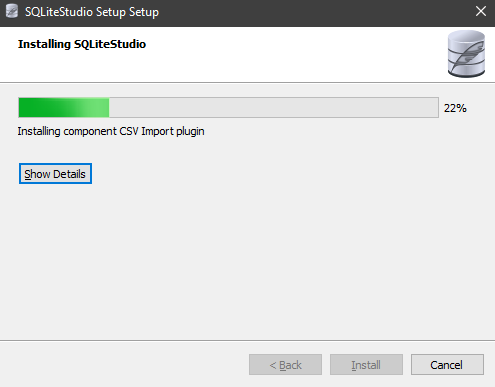
Step 1: Go to their website and click download current version for your chosen OS



Step 2: Follow the setup wizard form the downloaded file and accept prompts that pop up

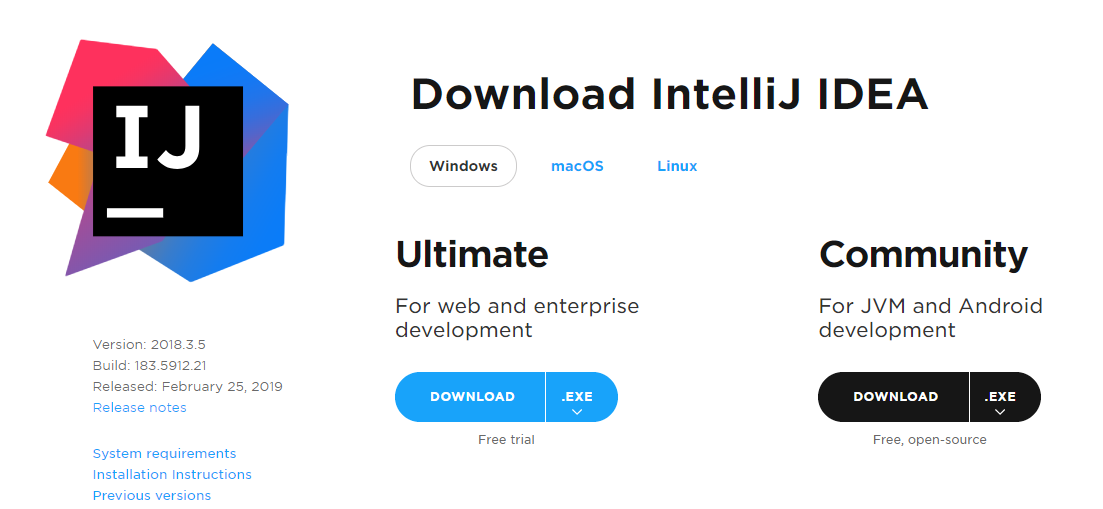


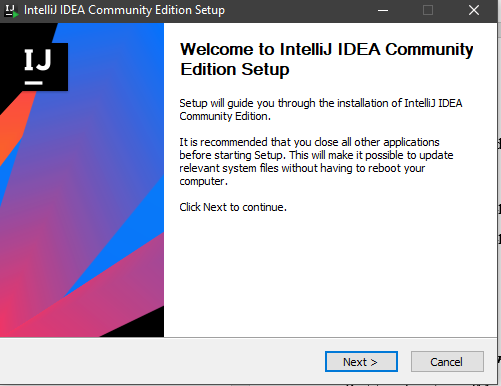
Step 3: Click install and now you can use the IDE for SQLite

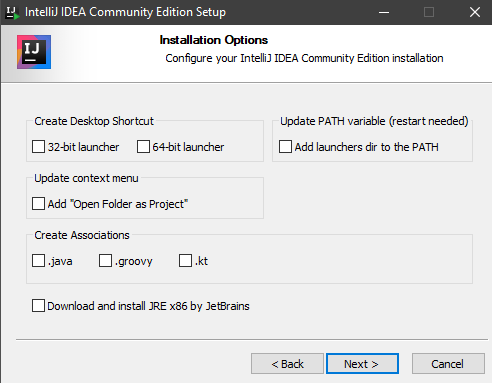


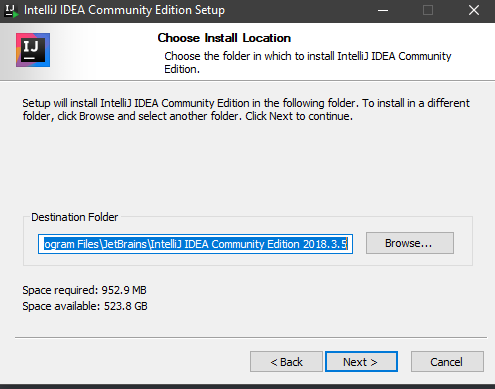
IntelliJ IDEA:

Step 1: Go to their website and click download current version for your chosen OS

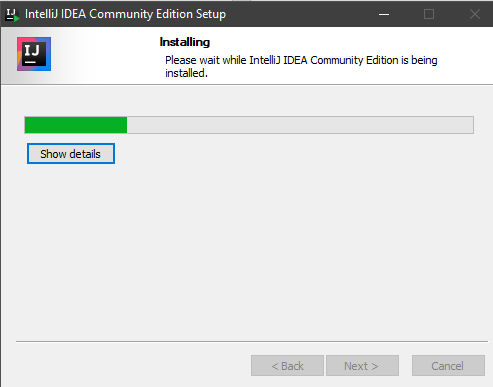


Step 2: Follow the setup wizard form the downloaded file and accept prompts that pop up





Step 3: Click install and now you can use the IDE for Java



B: Functionality Testing:

***Identify Features (Functionality):***

1. Login
2. Send Money

***Feature One, Login:***

Partition Input**:** login(String U, string P)

* String U is a string for the user’s Username
  + One possible partition is string with length < 0, string with length = 0, string with length > 0
* String P is a string for the user’s Password
  + One possible partition is string with length < 0, string with length = 0, string with length > 0

Test Specification:

* Username: <String a-z, A-Z, 0-9, special characters>
* Password: <String a-z, A-Z, 0-9, special characters>

Test Case:

*#1:*

* Inputs:
  + Username: admin
  + Password: admin
* Outputs:
  + Logs in

#2:

* Inputs:
  + Username:
  + Password:
* Outputs:
  + Please input your username! Please input your Password!

***Feature Two, Send Money:***

Partition Input: sendMoney(String ID, int N)

* String ID is a string for the recipient’s Username
  + One possible partition is string with length < 0, string with length = 0, string with length > 0
* double N is an integer between 0 and the user’s maximum balance
  + One possible partition is a number: < -**∞ -** 0.00,0.00-**∞**, 0.00-…\*>, …\* is the user’s maximum balance

Test Specification:

* Recipient’s ID: <String a-Z, A-Z, 0-9, special characters>
* Amount: <0…\*>, \* is the user’ maximum balance

Test Case:

Assume user has $10 in balance

*#1:*

* Inputs:
  + ID: bobross
  + Amount: $3.01
* Outputs:
  + Successfully sent $3.01 to bobross

#2:

* Inputs:
  + ID:
  + Amount: $3.01
* Outputs:
  + Please add a destination

*#3:*

* Inputs:
  + ID: bobross
  + Amount:$ -3.01
* Outputs:
  + You can only transfer money out of your account

*#4:*

* Inputs:
  + ID: bobross
  + Amount: $500
* Outputs:
  + You can’t transfer more money than you have

**2 Implementing the Test Cases:**

Login Functionality:

|  |  |
| --- | --- |
| Test ID | UserApiControllerIntegrationTest{} |
| Purpose of Test | This unit test is designed for the software to test our different input partitions for possible strings for username and password to receive the software’s output, this way we can see if certain special characters from our test specification would create a bug in the system |
| Test Environment | The environment used is IntelliJ IDE because we wrote our unit tests in Java platform with Windows OS |
| Test Steps | Tester must import the code from our github directory : backend/src/test/java/pw/wp6/avocado\_toast/  api/UserApiControllerIntegrationTest.java  into a Java IDE to proceed running the test |
| Test Input | 1: username: admin password: admin  2: username: password:  3: username: kasgponaw password: kasdio |
| Expected Result | 1: user is logged in  2: String prompt: Please input your username!  Please input your password!  3: User is denied entry |
| Likely Problems/Bugs Revealed | Vulnerability to SQL injections in username and password field to bypass user authentication  Vulnerability to SQL injections in username and password field to data manipulation (inserting, deleting, changing data, etc) |

Send Money Functionality:

|  |  |
| --- | --- |
| Test ID | TransactionInput{} |
| Purpose of Test | This unit test is designed for the software to test our different input partitions for possible strings for recipient ID and doubles for transaction amount to receive the software’s output, this way we can see if certain recipient IDs or transaction amount could bug the system |
| Test Environment | The environment used is IntelliJ IDE because we wrote our unit tests in Java platform with Windows OS |
| Test Steps | Tester must import the code from our github directory: backend/src/test/java/pw/wp6/avocado\_toast/ api/ LedgerApiControllerIntegrationTest.java into a Java IDE to proceed running the test |
| Test Input | 1: ID: caradelvignee Amount: $10.10  2: 1: ID: Amount: $9.30  3: 1: ID: caradelvignee Amount: $-5.20  4: 1: ID: caradelvignee Amount: $500.00 |
| Expected Result | 1: Successfully sent $10.10 to caradelvignee  2: Please add a destination  3: You can only transfer money out of your account  4: You can’t transfer more money than you have |
| Likely Problems/Bugs Revealed | Vulnerability to SQL injections in recipient ID field to data manipulation (inserting, deleting, changing data, etc)  User could enter a negative double for amount and an amount exceeding their balance |

**3 Test Documentation:**

|  |  |  |  |
| --- | --- | --- | --- |
| Bug | The test uncovered the bug | Description of the bug | Action was taken to fix the bug |
| Vulnerability to SQL injections to bypass user authentication | Login | A user could bypass our authentication software and enter the website without a valid username and password | Change the permission and privileges so user does not have privilege to login if its invalid |
| Vulnerability to SQL injections in username, password, and recipient ID field to data manipulation (inserting, deleting, changing data, etc) | Login  Send Money | A user could bypass manipulate our data and insert, delete, or update the data in the username, password, and recipient ID field | Change permission and privilege so user does not have privilege to change data |
| User could enter a negative double for amount and an amount exceeding their balance | Send Money | A user could enter a negative double or an amount they do not have, and this would mess up our data tables with users having an amount they should not have | We added error prompts if the user would enter a negative number or an amount exceeding their balance, and they would be prompted to change the value |