AVOCADO TOAST: AN ONLINE

BANKING PLATFORM

DAMH PHAM ANANDITA DUBEY FLAVIU TAMAS

CARLOS DELEON ALEX PETROS

CONTENTS

1. Planning Scheduling and Peer Evaluation 2
2. Communication and Collaboration 2
   1. To-do List………………………………………………………………………………….2
   2. Screenshot………………………………………………………………………………..2
3. Revise and Refine your System 3
4. System Modeling
   1. Architecture modeling
   2. Behavioral modeling
5. Implementation
   1. Implement the Database Design
   2. Implement the Class Diagram Design
6. Testing
   1. Creating Testing environment and Test Cases
   2. Implementing the Test Cases
   3. Test Documentation
7. Report

LIST OF FIGURES

Figure 1   
Figure 2

1. Planning Scheduling and Peer Evaluation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Assignee | Email | Task | Duration | Depends on | Due date | Evaluation |
| Anandita | Adubey2 | System modeling | 2h | Use cases Class Diagrams | 03/15/19 | 100% completed the task |
| Alex | Apetros1 | Revise and refine your system | 2h | System requirements  System modeling | 03/15/19 | 100% completed the task |
| Carlos | Cdeleon1 | Planning Scheduling  Communication Collaboration | 2h | None | 03/15/19 | 100% completed the task |
| Damh | Dpham16 | Testing | 2h | Implementation | 03/15/19 | 100% completed the task |
| Flaviu | Ftamas1 | Implementation | 2h | Uses cases  Previous imp. | 03/15/19 | 100% completed the task |

1. Communication and Collaboration

A screenshot of a cell phone

Description automatically generated

1. Revise and Refine your System  
   1. ***What is your product, on a high level?***

* Our product is an online banking application, which will allow the customers of a bank or other another financial institution to perform money-management without physically visiting the bank.
* It will allow the customers to open their accounts, manage them electronically, to monitor them, to make transactions, pay their bills, transfer money, make deposits, and so on.
  1. ***Whom is it for?***
  + An online banking application is beneficial to everyone but is especially useful for those people with a stringent work schedule.
  + It will help them to manage their accounts and keep track of their activities in a quick manner with minimal costs without the need to visit a physical bank during working hours or make a phone call.
  1. ***What problem does it solve?***
* 24/7 availability, saving the customer from rushing the banks during working hours. With an online banking system, people can perform their tasks at a time that suits their work schedule.
* Stringent schedules, where customers with strict working hours to perform their banking activities effectively and conveniently.
* Centralized source of information, where instead of visiting different officials specialized in different tasks, the customer can use an online banking application flexible single enough to do any task in a click. Human bankers are not always available, but the application always will be, meaning there is no need to rush to the bank to get things done.
* Remain informed: With the online banking system, customers can easily receive up-to-date information regarding their upcoming deadlines or dues through notifications, emails, or text messages.
* Easy bill payments, so that there is no need to rush to the bank. Everything can be done at home instead.
  1. ***What alternatives are available?***
     + Services provided by 3rd party application, like Mint A phone app
     + In person banking services
     + Phone calls
     + ATMs
  2. ***Why is this project compelling and worth developing?***
     + It would reduce costs for banks by reducing the amount of human labor re-quired.
     + An electronic banking system would enable banks to keep stringent records
     + A computerized ledger would reduce the number of mistakes when calculating interests, making transactions, and so on.
     + It would increase customer satisfaction because they would have access to our banks from any place with Wi-Fi.
     + Providing flexibility to the customers to get their work done at minimal or no cost.
     + Saving time of the customers (customer need not necessarily visit the bank physically)
  3. ***Describe the top-level objectives, diﬀerentiators, target customers, and scope of your product.***
     + Objective: To create a product that performs to our requirements and have it sustainable for multiple lifecycles.
     + Differentiators: Our system won’t sell data to any third parties, have no hidden transaction fees, and will not participate in predatory lending practices
     + Target customers: Our target customer is the average citizen, anyone that currently uses or will use a bank for their transaction needs. (Basically useful to everyone, no matter what)
     + Scope: The scope of our project is building an online business ledger, making a database to store our information, developing a web app and making a UI for our customers.
  4. ***What are the competitors and what is novel in your approach?***
* Our competitors are Finacle, nCino, Oracle, etc. What we bring new to the table is that our system will be entirely online.
* Our platform is customer focused and will be fully transparent. We won’t sell customer data or charge hidden fees. Additionally, providing the customers with the flexibility to perform the tasks at the minimal or no cost.
  1. ***Make it clear that the system can be built, making good use of the available resources and technology.***
* Our system is possible because it will be very simplistic and direct. Clients will connect to our application software where they will be greeted by a user-friendly interface, which will primarily be coded in HTML, with CSS used to create a beautiful design for our clients.
* Finally, the frontend of our application will be connected to a backend SQL database. The database will be responsible for recording transactions such as withdrawals, deposits, bill pays, and more.
  1. ***What is interesting about this project from a technical point of view?***

The project will use a client-server architecture in order to be accessible to clients from anywhere. Emphasis will also be placed on graphical design, using our creativity to design a nice interface for our clients. Providing flexibility to the clients to perform the non-transactional tasks through online banking. That might include:

* + - Viewing account balances
    - Viewing recent transactions
    - Downloading bank statements
    - Downloading periodic account statements
    - Funds transfer between the customer’s linked accounts Paying third parties, including bill payments
    - Credit card applications
    - Register utility billers

1. System Modeling
   1. Architecture Modeling  
      A close up of a map

      Description automatically generated


* 1. Withdrawing Cash (sequence)A close up of a map

     Description automatically generated
  2. Depositing Cash(sequence)A screenshot of a cell phone

     Description automatically generated

1. Implementation
   1. For the backend, open up a console in ‘backend’ and run ‘gradlew.bat run’
   2. For the front end open up a console in frontend and run ‘npm install; npm start’
   3. The actual site is <https://avocado-toast.wp6.pw/>
   4. The username is admin and password is admin
2. Testing

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

Environment Test Framework:

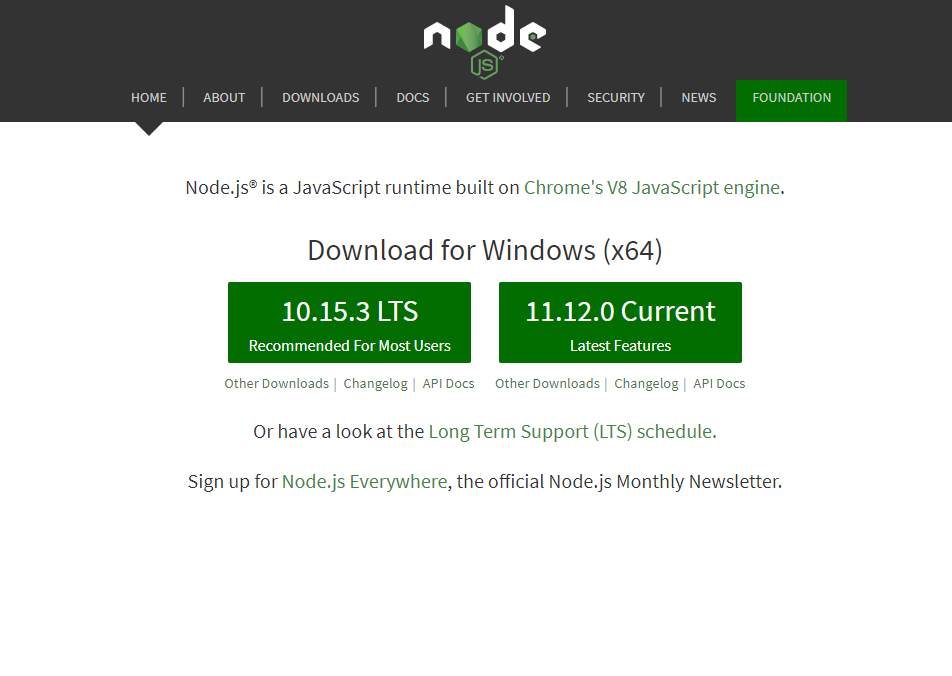
In this framework we decided to use:

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

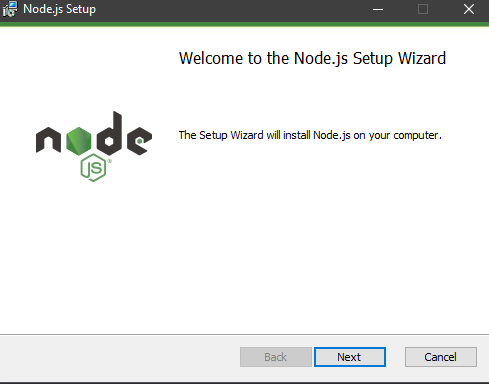
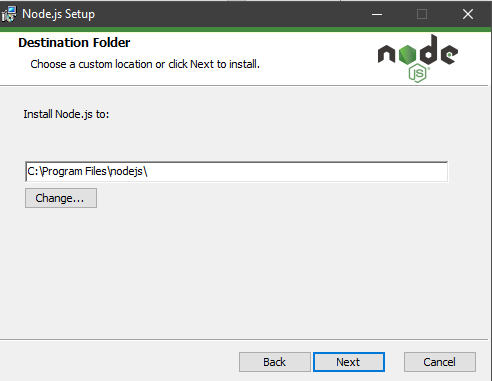
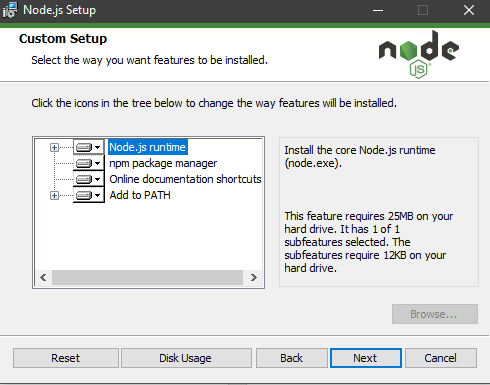
A: Installation of Framework:

*NodeJS:*

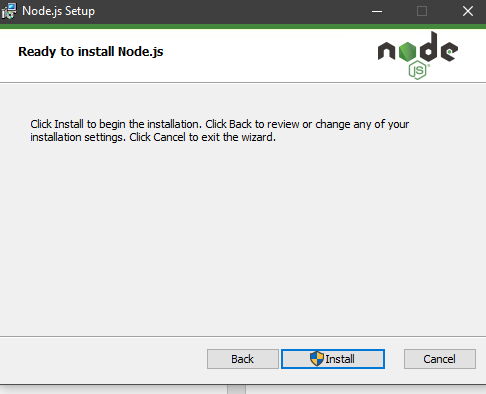
Step 1: Go to their website and click download current version for your chosen OS



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

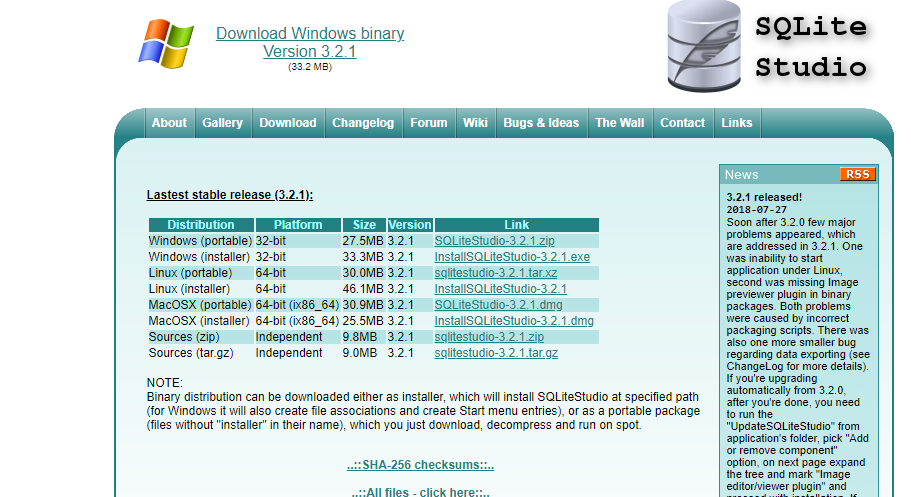


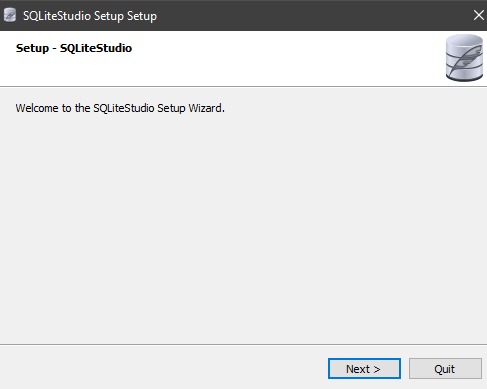
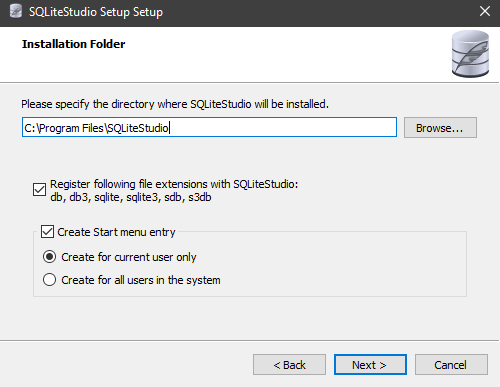
Step 3: Click install and now you can use the IDE for JavaScript

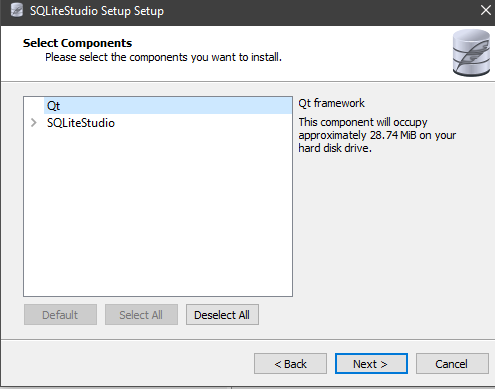


*SQLite:*

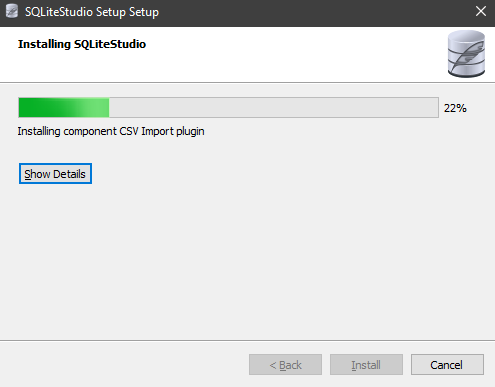
Step 1: Go to their website and click download current version for your chosen OS



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

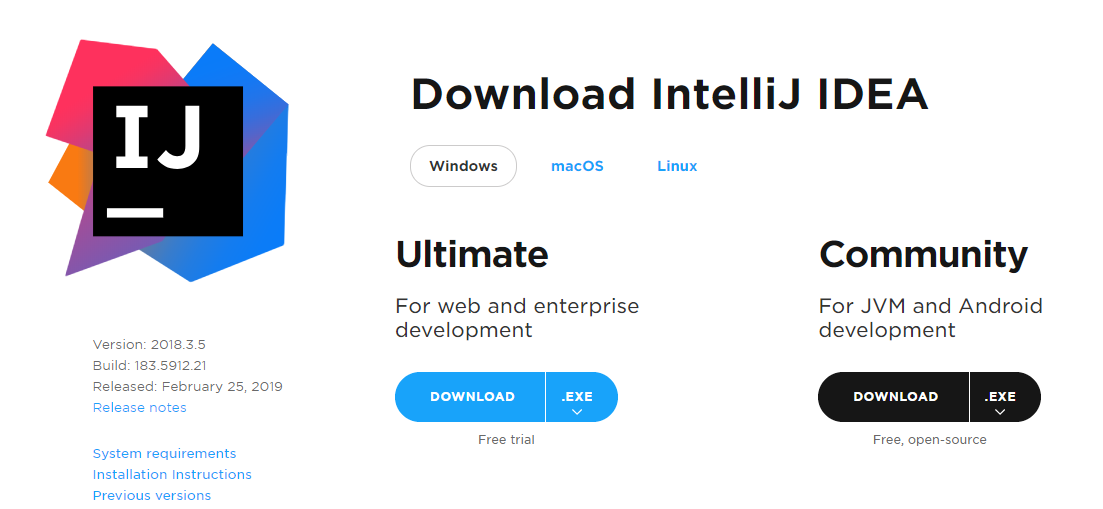


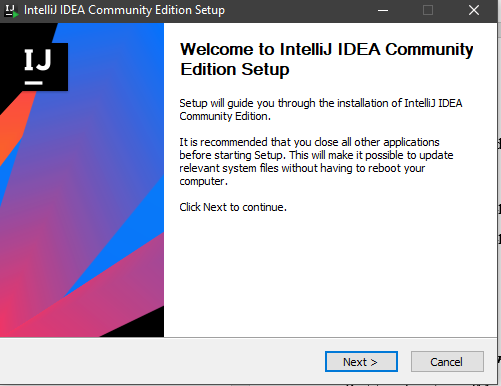
Step 3: Click install and now you can use the IDE for SQLite

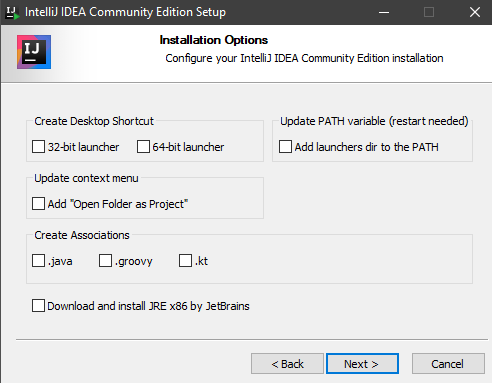


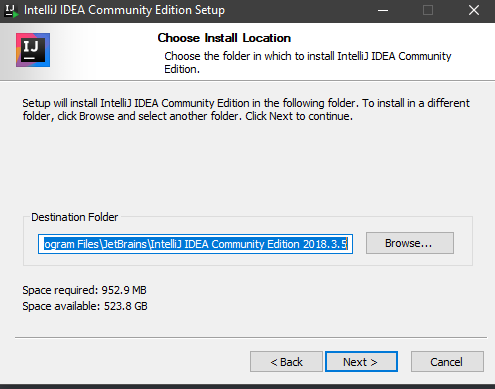
IntelliJ IDEA:

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

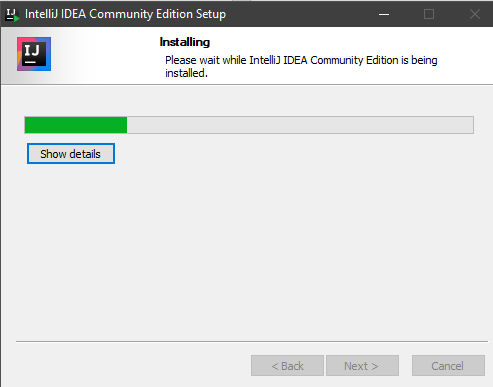


Step 2: Follow the setup wizard form the downloaded file and accept prompts that pop up





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 |