1.На модульном тестировании я бы проводил 2 глобальные проверки:

Тестировку вёрстки и функциональное тестирование

Тестирование вёрстки

Этап 1. Визуальная часть

На этом этапе необходимо проверить:

- Точность соответствия макета;
- Сетку;
- Масштаб;
- Изменение размера текстового поля;
- Выделение полей в фокусе;
- Верстку в разных разрешениях.

Этап 2. Доступность

На этом этапе необходимо проверить:

- Наличие нужных частей;
- Размер частей и их цвет;
- Выделяется ли текст в текстовых блоках;
- Нажимается ли иконка логотипа;
- Нажимаются ли кликабельные элементы;
- Расположение элементов относительно друг друга;
- Приобретает ли курсор разную форму при наведении на кликабельные элементы.

Этап 3. Корректная работа

На этом этапе необходимо проверить:

- Величину элемента;
- Корректность шрифтов;
- Цвета интерфейса;
- Контент;
- Фавикон;
- Стандарты HTML/CSS;

Функциональное тестирование

Этап 1.

Анализ исходных данных (технических требований) и согласование плана тестирования, тест-кейсов, сроков выполнения проекта, числа итераций;

Этап2.

Проведение функционального тестирования по тестовым сценариям с занесением выявленных багов в систему багтрэкинга;

Этап 3.

Составление отчета о проведенном тестировании и предоставление рекомендаций по улучшению системы.

На интеграционном тестировании я бы провёл 2 основные проверки:

Компонентную интеграционную проверку и системно интеграционную проверку

Компонентная интеграционная проверка

Проверяется взаимодействие между компонентами системы после проведения компонентного тестирования.

Системная интеграционная проверка

Проверяется взаимодействие между разными системами после проведения системного тестирования.

Системное тестирование

На системном уровне я бы сначала создал тестовый план, затем тест и юз кейсы, потом я бы создал тестовые данные и провёл бы автоматизированную проверку тест кейсов, а затем обычную. По результатам тестирования сформировал бы отчёт по выявленным ошибкам и провёл регрессионное тестирование после их исправления.

Приёмочное тестирование

В приёмочном тестировании я бы сначала определил бы какие модули нуждаются в доработке, затем я бы выбрал подходящий фреймворк для запуска тестов и приступил бы к написанию тестов.

2.Я бы провёл пользовательское приемочное тестирование.

1. Подготовлю план работ и ознакомлю с ним все стороны
2. Подготовлю всю информацию теста заранее
3. Подготовлю тестируемую среду
4. Дам пользователям контакты лиц ответственных за поддержку
5. Буду давать пользователям информацию о статусе тестирования
6. Подготовлю финальный отчёт

3. Первое, что бы я проверил в банковском приложении это вход в аккаунт.

2.Перевод денег с карты на карту

3.Ввод денег на карту и вывод

4.Добавление карты

5.Переход с карты на карту

4.

1.**Bottom up**

Все низкоуровневые модули, процедуры или функции собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования. Данный подход считается полезным, если все или практически все модули, разрабатываемого уровня, готовы. Также данный подход помогает определить по результатам тестирования уровень готовности приложения.

2. **Top down**

Вначале тестируются все высокоуровневые модули, и постепенно один за другим добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками с аналогичной функциональностью, затем по мере готовности они заменяются реальными активными компонентами. Таким образом мы проводим тестирование сверху вниз.

3. **Big Bang**

Все или практически все разработанные модули собираются вместе в виде законченной системы или ее основной части, и затем проводится интеграционное тестирование. Такой подход очень хорош для сохранения времени. Однако если тест кейсы и их результаты записаны не верно, то сам процесс интеграции сильно осложнится, что станет преградой для команды тестирования при достижении основной цели интеграционного тестирования.

4. **Hybrid**

Эта стратегия представляет собой комбинацию подходов «сверху вниз» и «снизу вверх». Здесь верхнеуровневые модули тестируются с нижнеуровневыми, а нижнеуровневые модули интегрируются с верхнеуровневыми, соответственно, и тестируются. Эта стратегия использует и заглушки, и драйверы.